トランジスタ技術 SPECIAL

特集 パソコン周辺機器インターフェース詳解

セントロニクス/RS-232C/GPIB/SCSIを理解するために

No.9

CQ出版社

エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン

トランジスタ技術 SPECIAL

季刊●B5判●定価：1～33定価1,570円 34～45定価1,631円 46～49定価1,723円 50～57定価1,835円 58以降定価1,840円

1 個別半導体素子　活用法のすべて
基礎からマスタするダイオード，トランジスタ，FETの実用回路技術

3 PC9801と拡張インターフェースのすべて
16ビット・パソコンを使いこなすためのハード&ソフト

4 C-MOS標準ロジックIC活用マニュアル
実験で作ぶ4000B/4500B/74HCファミリ

5 画像処理回路技術のすべて
カメラとビデオ回路，パソコンと隔合させる

6 Z80ソフト&ハードのすべて
基礎からマクロ命令を使いこなすまでのノウハウを集大成

8 データ通信技術のすべて
シリアル・インターフェースの基礎からモデムの設計法まで

9 パソコン周辺機器インターフェース詳解
セントロニクス/RS-232C/GPIB/SCSIを理解するために

10 IBM PC&80286のすべて
世界の標準パソコンとマルチタスクの基礎を理解する

11 フロッピ・ディスク・インターフェースのすべて
需要の急増するFDDシステムの基礎から応用まで

13 シミュレータによる電子回路理論入門
コンピュータを使ったアナログ回路設計の手法を理解するために

14 技術者のためのCプログラミング入門
MS-C，Quick C，Turbo Cによるソフトウェア設計のすべて

15 アナログ回路技術の基礎と応用
計測回路技術のグレードアップをめざして
〈在庫僅少〉

16 A-D/D-A変換回路技術のすべて
アナログとディジタルを結ぶ最新回路設計ノウハウ

17 OPアンプによる回路設計入門
アナログ回路の誤動作とトラブルの原因を解く

19 PC9801計測インターフェースのすべて
オリジナル拡張ボードでパソコンを実践活用しよう

20 アナログ回路シミュレータ活用術
ゲーム感覚の回路設計を体験しよう

22 ディジタル回路ノイズ対策技術のすべて
TTL/CMOS/ECLの活用法と誤動作/トラブルへの処方

23 回路デザイナのためのPLD最新活用法
PLDのプログラミング法からPALライタの製作まで

24 Cによる組み込み機器用プログラミング
16ビットCPUによるメカトロニクス入門
〈在庫僅少〉

27 ハードディスクとSCSI活用技術のすべて
本格活用のためのハード&ソフトのすべてを詳解

28 最新・電源回路設計技術のすべて
3端子レギュレータから共振型スイッチング電源まで

29 マイコン独習Z80完全マニュアル
手作りの原点から実用ソフトの作成まで

30 ニュー・メディア時代のデータ通信技術
赤外線，無線通信技術からLAN，光ファイバを用いた高速通信技術まで

31 基礎からのビデオ信号処理技術
複合映像信号の理解からハイビジョン信号の捉え方まで

32 実用電子回路設計マニュアル
アナログ回路の設計例を中心に実用回路を詳述

33 オプト・デバイス応用回路の設計・製作
光素子を使いこなすための製作ドキュメント

34 つくるICエレクトロニクス
機能ICを使って実用機器を作ろう
〈在庫僅少〉

35 C言語による回路シミュレータの製作
Quick Cでのプログラミングとフィルタ回路の解析

36 基礎からの電子回路設計ノート
トランジスタ回路の設計からビデオ画像の編集まで

37 実用電子回路設計マニュアルⅡ
豊富な回路設計例から最適設計を学ぼう

38 Z80システム設計完全マニュアル
周辺I/Oボードの設計とマイコン・システムの開発

39 A-Dコンバータの選び方・使い方のすべて
アナログ信号をディジタル処理するための基礎技術

40 電子回路部品の活用ノウハウ
機器の性能と信頼性を支える受動部品の使い方

41 実験で学ぶOPアンプのすべて
汎用OPアンプから高性能OPアンプまで

42 高速ディジタル回路の測定とトラブル解析
ハイスピード・ディジタル信号を高周波と捉らえる

43 Cによるマイコン制御プログラミング
86系ペリフェラルを中心とした

44 フィルタの設計と使い方
アナログ回路のキーポイントを探る

45 PC98シリーズのハードとソフト
386&486マシンを使いこなす！

46 アナログ機能ICとその使い方
民生用AV機器からマルチメディア分野で活躍する

47 高周波システム&回路設計
通信新時代の回路技術とシステム設計

48 作れば解るCPU
ロジックICで実現するZ80とキャッスル・マシン

49 徹底解説　Z80マイコンのすべて
Z80CPUの概要から周辺LSIの活用法，ICEのデバックまで

50 フレッシャーズのための電子工学講座
電磁気学の基礎から電子回路の設計，製作までをやさしく解説

51 データ通信技術基礎講座
RS232Cの徹底理解からローカル通信の実用技術まで

52 ビデオ信号処理の徹底研究
映像信号の基礎から高画質化のためのディジタル信号処理の方法まで

53 パソコンによる計測・制御入門
研究室や実験室で必要なデータ収集のノウハウを基礎から解説

54 実践パワー・エレクトロニクス入門
パワーMOS FETとIGBTの使い方をやさしく解説

55 作ってわかる電子回路製作入門
やさしい電子工作からパソコンを使ったシステム開発まで

56 電子回路シミュレータ活用マニュアル
アナログ回路解析だけでなくディジタル回路解析も追加された

57 最新・スイッチング電源技術のすべて
効率とノイズを重点的に解説したソフト・スイッチングの指南書

58 基本・C-MOS標準ロジックIC活用マスタ
低電圧動作とドライブ能力の向上をはかった

CQ出版社 〒170 東京都豊島区巣鴨1-14-2 販売部☎03-5395-2141 振替 00100-7-10665

定価は1997年4月1日現在の消費税5%を含んだ表示です

トランジスタ技術 SPECIAL No.9

CONTENTS

FEATURES

セントロニクス／RS-232C／GPIB／SCSIを理解するために

パソコン周辺機器インターフェース詳解

セントロニクス・パラレル

RS-232Cシリアル

GPIB

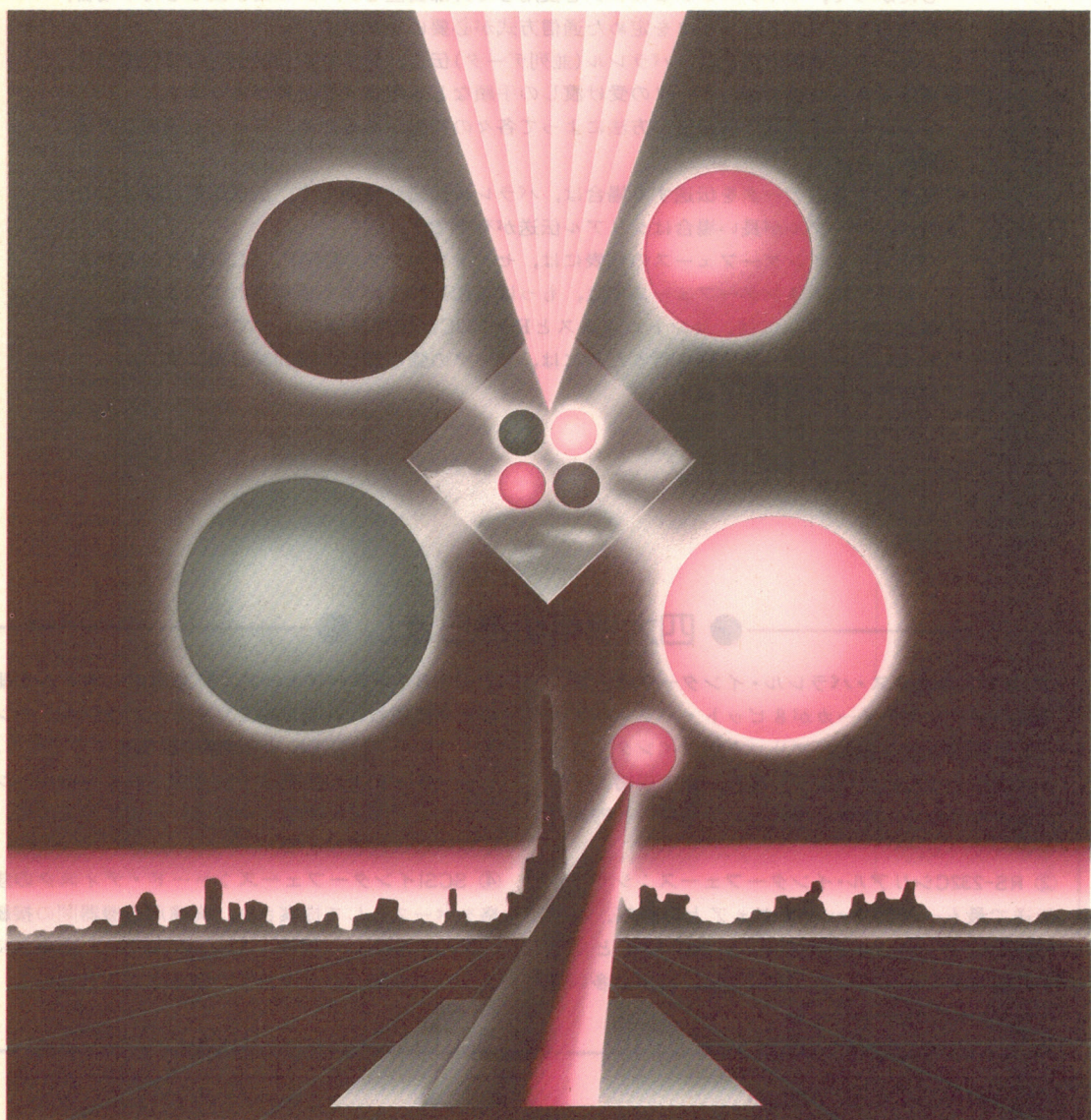
SCSI

表紙デザイン：(株)シーピーユー　簑原圭介

FEATURES

現在，パソコン/マイコンの周辺機器とのインターフェースとして，セントロニクスやRS-232C，GPIBが多く使われ，将来は高速データ転送が可能なSCSIも利用されそうです．本誌では，それぞれの規格と使われるICの説明を行ったのち，実際の応用例を示し理解を深めます．

セントロニクス/RS-232C/GPIB/SCSIを理解するために

パソコン周辺機器インターフェース詳解

● はじめに

私たちが，他人と会話する場合，お互いの話を確認しながら会話を進めていきます．もしも，同時に話した場合，お互いになにを言っているのかわからず，会話にならなくなります．このようなことが，コンピュータと外部装置の間でも起こります．

コンピュータと外部装置では，次のような会話になります．

コンピュータ：「データを送ります．いいですか．」
外部装置：「データを送ってください．」
コンピュータ：「データを送っています．」
外部装置：「データを受け取りました．」

となります．

この一連の手順を，ハンドシェイクと呼びます．

したがって，マイクロコンピュータを使用して外部装置とデータの受け渡しを行う場合，互換性のあるハンドシェイクを定めた通信方式が必要になります．

また，その通信方式には，パラレル(並列データ)伝送，シリアル(直列データ)伝送の2種類が考えられますが，データの受け渡しの手順なども考慮する必要があります．

ハンドシェイクは，各伝送の方法によって各々の特徴があるため，もっとも適切な方法を選択します．

高速で大量のデータを伝送する場合は，パラレル伝送が有効です．伝送速度は遅くなりますが，ケーブルが長い場合はシリアル伝送が有効です．

パラレル・インターフェースの代表には，セントロニクス，GBIP，SCSIなどがあります．シリアル・インターフェースでは，もっぱらRS-232Cがよく使用されています．

最近のパソコンには，セントロニクスとRS-232Cが標準装備になっています．実際，これらによってデータ伝送を行うためには，データの受け渡しの規則が必要です．

以下それらの説明をします．

● 四大規格のあらまし ●

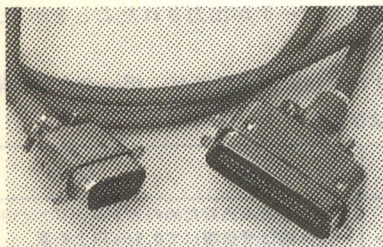
① **セントロニクス・パラレル・インターフェース**：米国のプリンタ・メーカが8ビット・パラレル・データに制御信号を加え，プリンタ用のインターフェース規格としたのが始まり．現在，標準規格と呼ばれるものはなく，各社は準拠という独自の社内規格を作っている．

② **RS-232Cシリアル・インターフェース**：シリアル伝送で最も一般的なハードウェア上の規格．現在EIA-232-Dという名前になって，RS-232Cとは若干の変更がなされている．本誌では，利用の多い従来からのRS-232Cを用いる．

③ **GPIBインターフェース**：計測用のシステムを組むときに使われる双方向8ビット・パラレル・インターフェース．IEEE-488が標準規格となっており，ハードウェア上の接続は十分考慮されているが，ソフトウェア面で利用者により違いが出てくるので，互換性には注意が必要．

④ **SCSIインターフェース**：ハード・ディスクの規格をベースとして提案され，高速周辺機器間の接続に使われる新しい規格．ANSIのX3T9.2 委員会により，より細かいコマンドの制定もなされている．

§ 1-1 セントロニクス・インターフェースの基礎

里 和政

セントロニクス・インターフェースは，米国セントロニクス社が自社のプリンタ用に開発したコンピュータからプリンタにデータを送るための規格で，安価でかつ高速のデータを送ることができます．現在のプリンタは，ほとんどこのセントロニクスが標準となっています．

このインターフェースの信号線は，すべてTTLレベルで入出力されています．基本的な制御は，3本で行われ，低速または高速にかかわらずどのようなプリンタでも，CPUと同期を取ることができます．

● セントロニクス・コネクタのピン配置

図1に，セントロニクス・コネクタのピン配置を示します．基本的な端子は，1ピンから11ピンの信号線です．これらの線は，すべてツイスト・ペア(19ピンから29ピンのグラウンド・ピンとのより線)になっています．

ほかの端子については，各社で適当な信号線を定義しています．

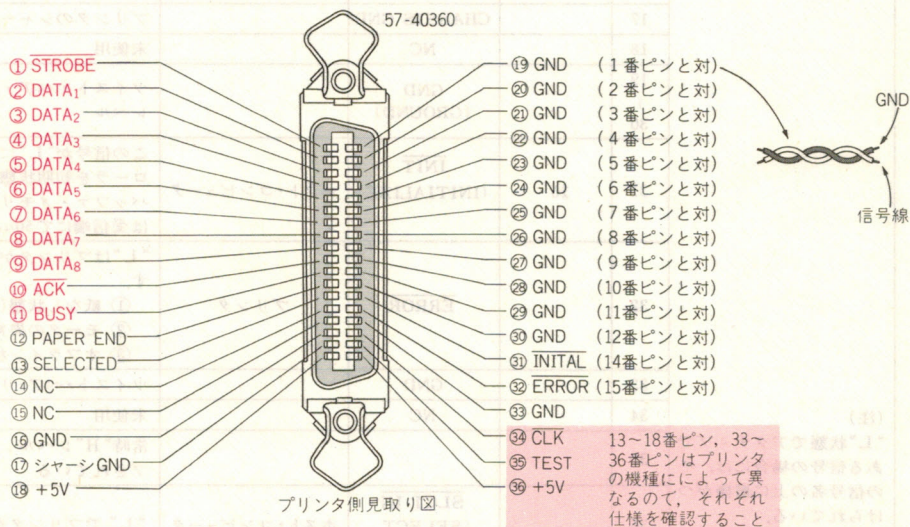
例として，スター精密，エプソン，日本電気のPC-PR系の各社のプリンタについての各ピンを表1，表2，表3に示します．

セントロニクスのタイミング

セントロニクスのデータ伝送は，DATA STROBE, ACKNOWLEDGE(ACK), BUSYの3本の制御線と

〈表1〉[1] スター精密社AR2400のコネクタ・ピンの説明

ピン番号	信号名	入出力区分	ペア・リターン・ピン
1	$\overline{\text{STROBE}}$	入力	19
2 ∫ 9	$DATA_1$ ∫ $DATA_8$	入力	20 ∫ 27
10	$\overline{\text{ACK}}$	出力	28
11	BUSY	出力	29
12	PAPER END	出力	30
13	SELECTED	出力	
16	GND		
17	CHASSIS GND		
18	+5V	出力	
31	$\overline{\text{INPUT PRIME}}$	入力	
32	$\overline{\text{ERROR}}$	出力	
33	EXT GND		

〈図1〉セントロニクス・コネクタのピン配置

DATA線によって行います．まず最初に，制御方法について説明しますが，その前に各制御線の意味について，まとめておきます．

▶DATA STROBE
DATA線に，データが出力されたことを示す．

▶BUSY
現在プリンタが動作中であり，データを受け取れ

〈表2〉[2] エプソン社VP130Kのピン配置および説明

ピン番号	リターン側ピン番号	信号名	発信先	機能
1	19	$\overline{\text{STROBE}}$	ホスト・コンピュータ	データを読み込むためのストローブ・パルス．パルス幅は受信端にて0.5μs以上必要．定常状態では"H"であり"L"となった後にデータを読み込む
2 3 4 5 6 7 8 9	20 21 22 23 24 25 26 27	DATA$_1$ DATA$_2$ DATA$_3$ DATA$_4$ DATA$_5$ DATA$_6$ DATA$_7$ DATA$_8$	ホスト・コンピュータ	各信号はパラレル・データの1ビット目から8ビット目までの情報を表す．"H"はデータが1であり"L"はデータが0であることを示す
10	28	$\overline{\text{ACKNLG}}$ (ACKNOW-LEDGE)	プリンタ	"L"はプリンタが次のデータを受け付ける用意ができていることを示す．パルス幅は約11μs
11	29	BUSY	プリンタ	"H"はプリンタがデータを受け取れないことを示す．逆に"L"はプリンタがデータを受け取れることを示す．この信号が"H"になるのは次の場合， ① データ・エントリ中 ② 印字中およびヘッド・キャリア動作中の一部の時間 ③ 紙送り中の一部の時間 ④ エラー状態 ⑤ オフライン状態
12	30	PE (PAPER END)	プリンタ	"H"はプリンタに用紙がないことを示す
13		SLCT (SELECT)	プリンタ	常時"H"レベル．3.3kΩで＋5Vにプルアップされている
14		$\overline{\text{AUTO FEEDXT}}$	ホスト・コンピュータ	この信号が"L"になると，プリンタは印刷終了後，自動的に1行の改行を行う
15		NC		未使用
16		GND		ロジック供給電源の0Vレベル
17		CHASSIS GND		プリンタのシャーシのGNDレベル
18		NC		未使用
19 ≀ 30		GND (GROUND)		ツイスト・ペア・リターン用信号グラウンド・レベル
31	16	$\overline{\text{INIT}}$ (INITIALIZE)	ホスト・コンピュータ	この信号が"L"になると，プリンタ・コントローラを初期状態にリセットし，プリント・バッファ・メモリがクリアされる．パルス幅は受信端にて50μs以上必要
32		$\overline{\text{ERROR}}$	プリンタ	"L"はプリンタがエラー状態にあることを示す． ① 紙なし状態(ESC8で解除される) ② モータの異常動作 ③ オフライン状態
33		GND		ツイスト・ペア・リターン用グラウンド
34		NC		未使用
35				常時"H"レベル．3.3kΩで＋5Vにプルアップされている
36		$\overline{\text{SLCT IN}}$ (SELECT INPUT)	ホスト・コンピュータ	"L"でプリンタを選択する

パラレル・インターフェース信号の場合には，ツイスト・ペア線を使用しリターン側をロジカル・グラウンドに接続する．

(注)
"L"状態でアクティブである信号の場合には，その信号名の上に横棒がつけられている．

ないことを示す.
(または，データ・バッファがフルである)
▶ACK
データの読み取りが正常に終了した.

セントロニクスのタイミングも各社によって差があります. セントロニクス・インターフェースのタイミングを図2，図3，図4に示します.

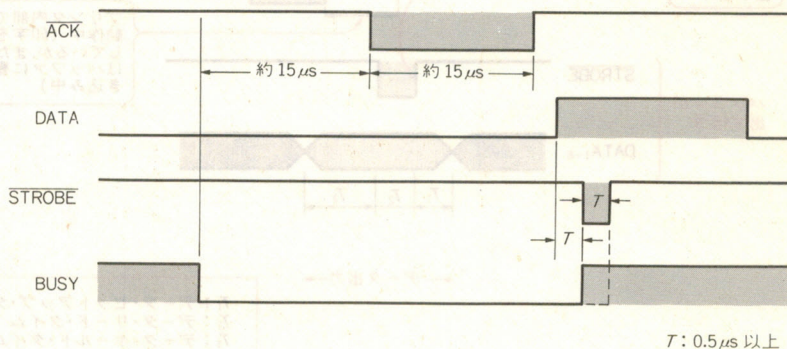
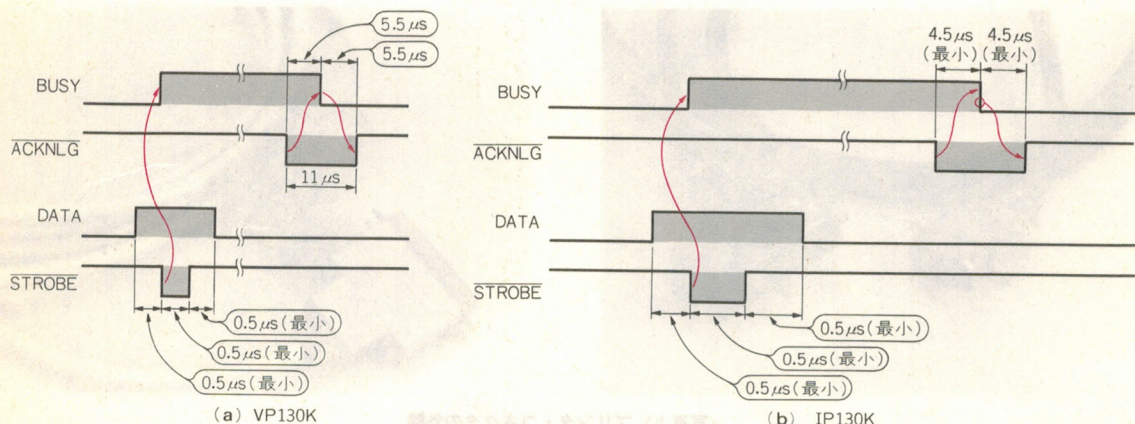
制御の基本的なタイミングは，図5のようになります. タイムチャートを見るとわかるように，BUSY信号と同じタイミングでACK信号もインアクティブとなるために，3線ハンドシェイクでなくても，2線ハンドシェイクで動作できます.

2線ハンドシェイクは，DATA STROBEとACKまたはBUSYのどちらか一方を使用します. 一般的には，DATA STROBEとBUSYがよく使用されています.

最近のパソコンでは，2線ハンドシェイクを採用しているものが多くあります. 参考例に，PC9801のセ

〈表3〉[3] 日本電気製PC-PR201H2のピン配置

ピン番号	信号名	入出力区分	ペア・リターンピン	備考
1	$\overline{\text{DATA·STB}}$	入力	19	
2 〜 9	DATA1 〜 DATA8	入力	20 〜 27	
10	$\overline{\text{ACK}}$	出力	28	
11	BUSY	出力	29	
12	PE	出力	30	Paper End (紙切れ)
13	SELECT	出力		プリンタのセレクト(オンライン)
16	SG			Signal Ground
17	FG			Frame Ground
18	＋5V	出力		
31	$\overline{\text{INPUT PRIME}}$	入力		初期化(プリンタを初期化する)
32	$\overline{\text{FAULT}}$	入力		エラー
33	SG			グラウンド

〈図2〉[1] スター精密製AR2400のタイムチャート

〈図3〉[2] エプソン製プリンタのタイムチャート

ントロニクス・コネクタの配置を図6に示します．この場合，ACK線には何も接続しません．

〈図4〉[3] 日本電気製PC-PR201H2/460LP/601のタイムチャート

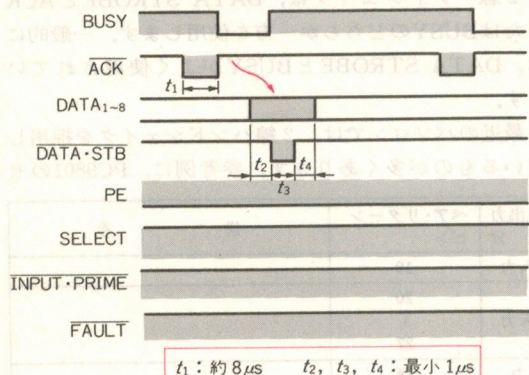
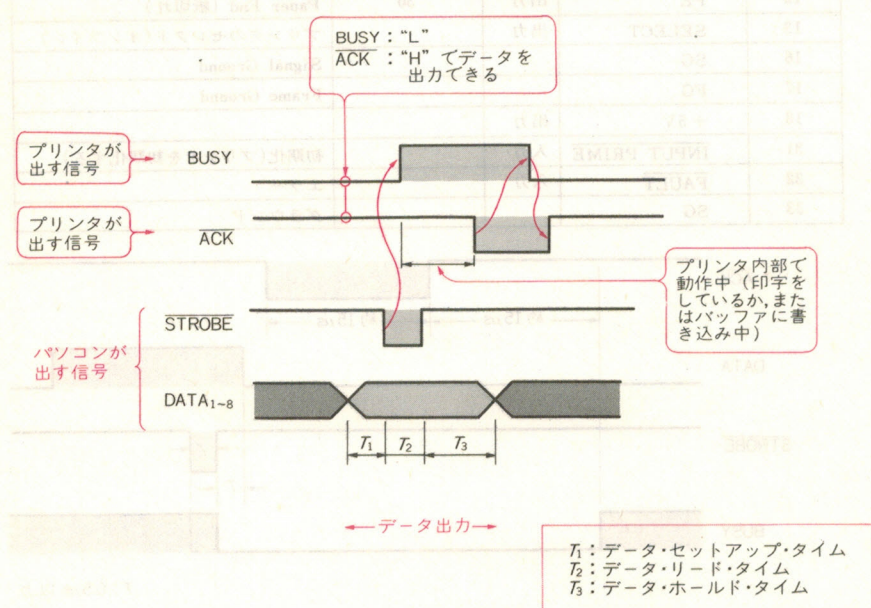
〈図5〉セントロニクスの基本タイムチャート

今回使用したセントロニクス・プリンタは，図1で示したコネクタのピンとは若干異なります．紙切れ(PE)，セレクト・ピンなどがありますが，これらの信号はとくに考慮する必要はなく，STROBEとBUSY信号があれば制御できます．

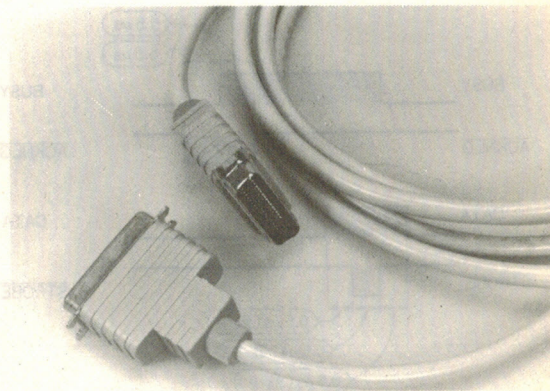
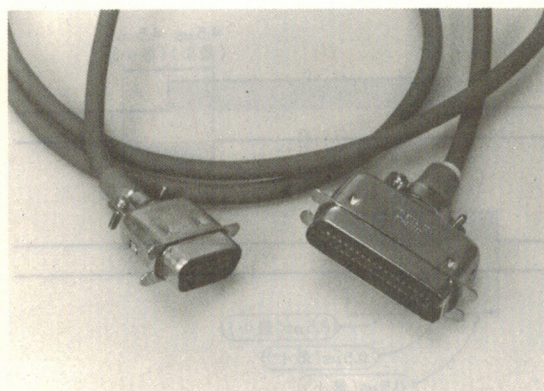
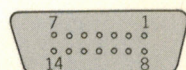
ただし，これらの信号を使用することにより，確実な制御が可能になります．

上記以外の制御線については，各プリンタの説明書を参照してください．

セントロニクスの短所は，CPU→プリンタと一方のみの伝送しか行えないことです．

制御用のLSIとしては，8255A，Z80 PIO，6821などがありますが，近ごろでは専用のLSIも各半導体メーカから発表されています．

〈図6〉PC9801のセントロニクス用プリンタ・インターフェースのピン配置

ピン番号	信号名
1	PSTB
2	PDB_0
3	PDB_1
4	PDB_2
5	PDB_3
6	PDB_4
7	PDB_5
8	PDB_6
9	PDB_7
10	NC
11	BUSY
12	NC
13	NC
14	GND

7　1
14　8

〈写真1〉プリンタ・コネクタの外観

用語解説

● **ツイスト・ペア**

プリンタのケーブルのように複数の信号線がある場合，お互いの線同士が干渉し合ってノイズが乗らないように，信号線ごとにグラウンド線をねじり合わせたものです．

● **プリンタ・データ・バッファ**

プリンタの印字速度は，CPUからのプリント・データの転送速度よりもかなり遅いために，CPUがプリンタを待つことになります．CPUはその間ほかの仕事ができないため，プリンタ自身に数10Kバイトのメモリをもたせて，プリント・データを先に取り込むようにしたものです．このメモリをプリンタ・データ・バッファと呼びます．

● **タイミング**

装置などを制御する場合に，各制御信号を有効にする時間間隔のことです．

● **インアクティブ**

制御線が“L”か“H”かを示すときに用いるが，通常アクティブを真(有効)としてインアクティブを偽(無効)とします．制御線が負論理の場合は，アクティブを“L”，インアクティブを“H”とします．正論理の場合は，その逆になります．

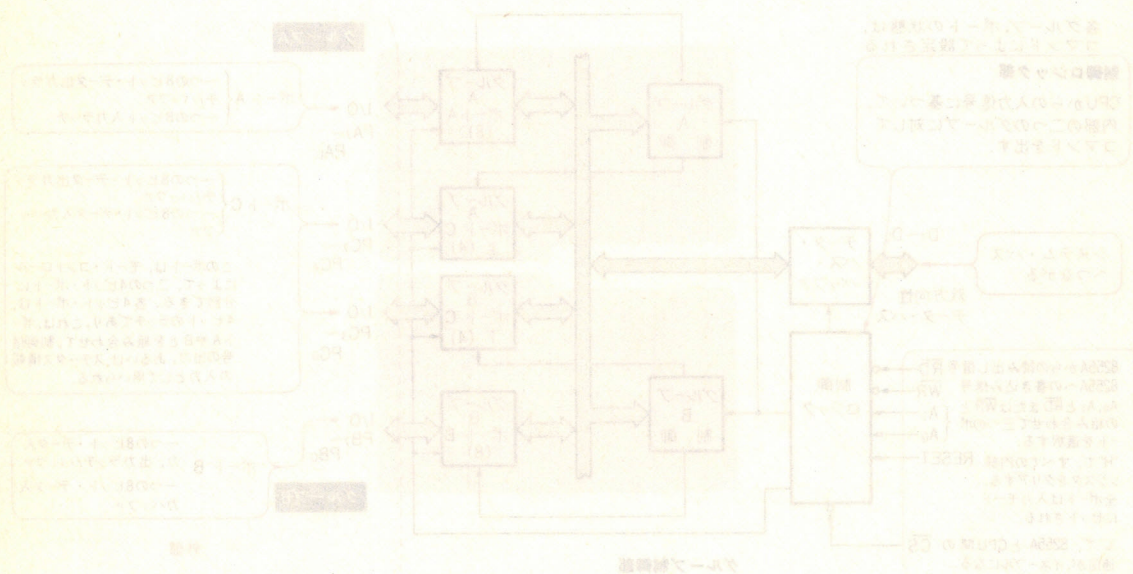
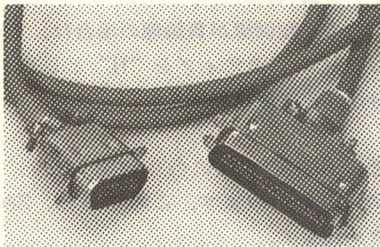
● **8080系**

インテル社の8080 CPU用に合うバス・タイミングで作られたLSIをいいます．この中には，Z80，8086なども含まれます．

● **I/Oウェイト**

CPUが，周辺LSIに対して入出力を行う場合に，LSIのI/Oタイミング(アクセス・スピード)が遅いために，アクセスが終わるまでCPUが待つことをいいます．

● **モード**

周辺LSIには，汎用性をもたせるため多くの機能があり，プログラムで指定することによって選択することができます．この機能のことをモードと呼んでいます．

§1-2 パラレル・インターフェース用LSIの使い方

里 和政/神崎康宏/斉藤健司

8255A(PPI)

8255Aは，8080系のCPUでよく利用される，プログラマブル・パラレルI/Oコントローラであり，8/16ビットCPUの区別なしに使用されています．32ビットCPUでも使用されるでしょう．

このLSIは，プログラムによってコントロールすることができます．

● 8255Aの内部構成

8255Aの内部ブロック図を図1に，図2に端子配置図を示します．I/Oポートは，A,B,Cと三つあります．A,B,Cと三つに分けると同時に，AとCの半分，BとCの半分という，二つのグループに分けて考えることもできます．各モードに応じて，A,Bはそれぞれ独立して機能します．Cは，モード0以外は4ビットずつA,Bのそれぞれのグループとともに，コントロール信号としての機能を果たします[7]．

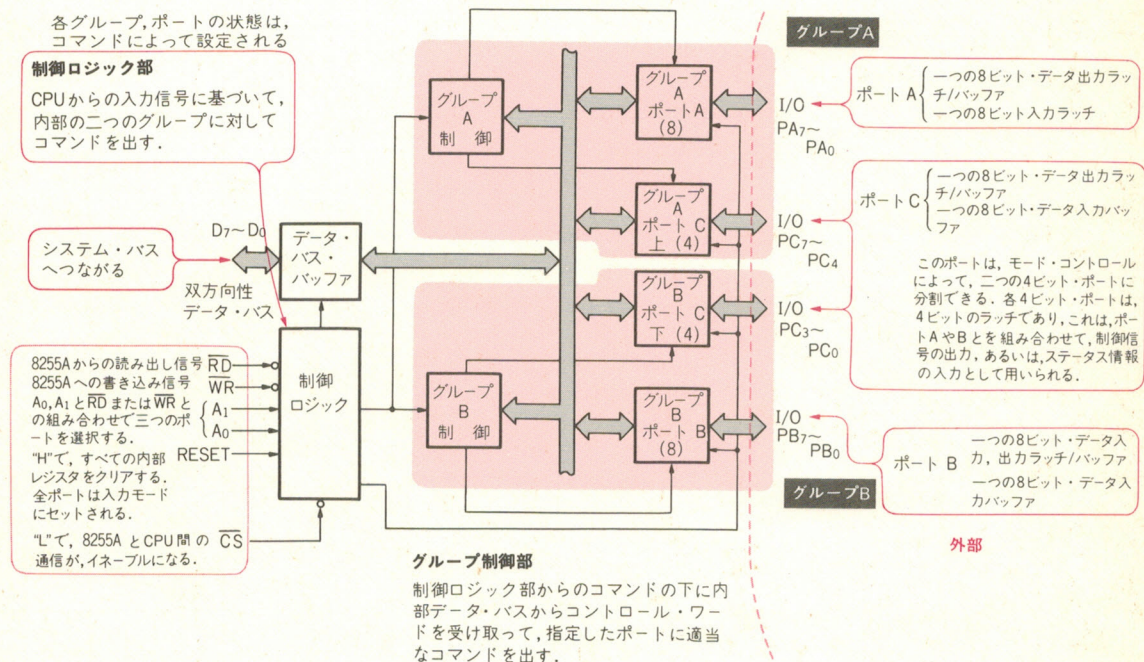
● モードの使い方

この素子の最もシンプルな使用方法は，モード0のたんなる入出力のみを行うポートとして利用する場合です(図3，図4)．

この場合，入力動作ではその入力時にA,B,Cの各ポートに加わっているデータがCPUに読み込まれます．出力動作では，各ポートへ出力されたデータは，出力動作後も同じデータが出力され続けます．これは，各ポートへの書き出し時に出力データが保持されるためです[7]．

● 8255Aのタイミング上の問題点

本来このLSIは，8ビット用に設計されているために，16/32ビットなどの高速のCPUに対しては，アクセス・タイムなどで問題が発生することもあります(高

〈図1〉[7] 8255Aのブロック図

速用のLSIもある).

そのため，CPUがI/Oをアクセス時に，I/Oウェイトを入れる必要があります．また，リセット直後，I/O素子にコマンド・ライト後，内部動作を行うために，若干のソフトによるウェイトが必要です．

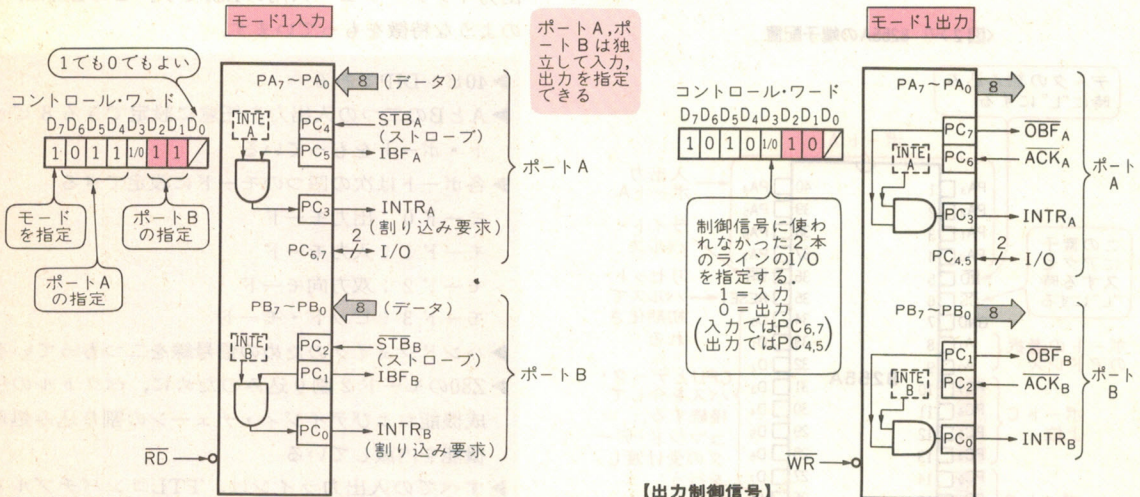
これらの点については，マニュアルをよく読むことです．実際使ってから，わかることもあります．

◈ セントロニクスへの応用

この8255Aを使用して，セントロニクス・インターフェースを考えてみます．ハンドシェイクは，2線式とします．

● モードの設定手順

ここでの8255Aのモードは，モード0を使用します．

〈図3〉[7] 8255Aモード1の説明

【入力制御信号】

$\overline{STB}$(Strobe Input):
　このピンへの"L"入力データが入力ラッチへ入る．

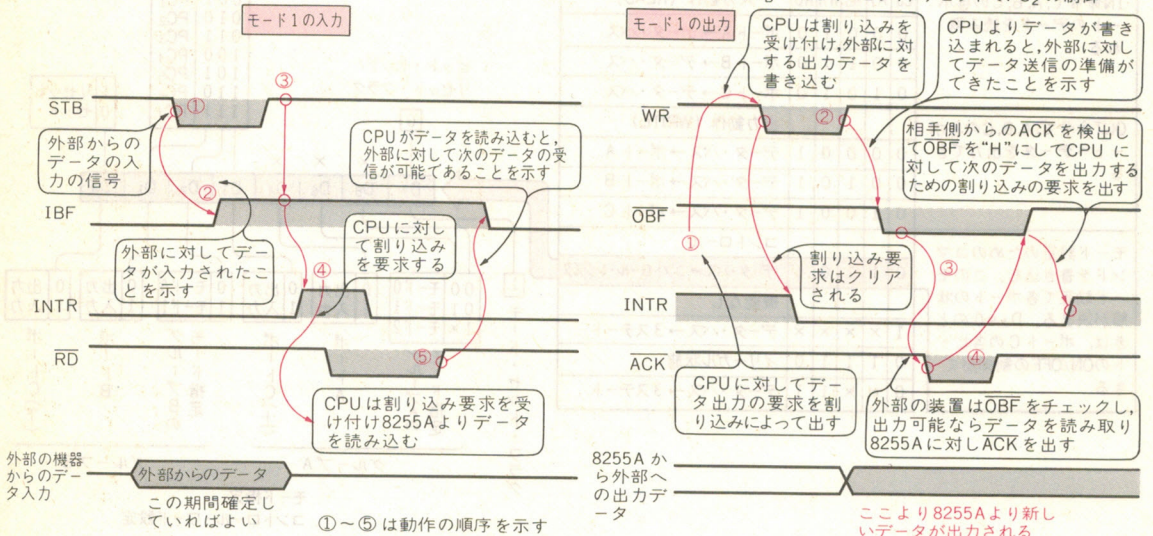
IBF(Input Buffer Full FF):
　$\overline{STB}$に対する応答信号．データが入力ラッチに入ったことを"H"で示す．$\overline{STB}$が"L"になることでセットされ，$\overline{RD}$入力の立ち上がりでリセットされる．

INTR(Interrupt Request):
　CPUに対する割り込み要求信号として用いるもので，"H"になる．$\overline{STB}$＝1，IBF＝1，INTE＝1のときセットされ，$\overline{RD}$の立ち下がりでリセットされる．
　内部割り込みマスク・フリップフロップの制御は，
　▶$INTE_A$―ビット・セット/リセットによるPC_4の制御
　▶$INTE_B$―ビット・セット/リセットによるPC_2の制御

【出力制御信号】

$\overline{OBF}$(Output Buffer Full FF):
　CPUが指定したポートにデータを書き込んだとき，このピンが"L"になる．このフリップフロップ(FF)は，$\overline{WR}$入力の立ち上がりでセットされ，$\overline{ACK}$が"L"になることでリセットされる．

$\overline{ACK}$(Acknowledge Input):
　CPUが出力したポートAまたはポートBのデータをI/O側で受け取ったとき，8255Aに対してその応答信号である"L"を出力する．

INTR(Interrupt Request):
　CPUに対する，I/Oからのデータ転送終了割り込み信号で"H"が出る．この信号は，$\overline{OBF}$＝1，INTE＝1のときに，$\overline{ACK}$によってセットされ，$\overline{WR}$の立ち下がりでリセットされる．
　内部割り込みマスク・フリップフロップの制御は，
　▶$INTE_A$―ビット・セット/リセットでPC_6の制御
　▶$INTE_B$―ビット・セット/リセットでPC_2の制御

データの出力用にはAポート，BUSY信号の入力用にCポートの下位4ビットのうちの1ビットを割り当てます．STROBE信号の出力用にCポートの上位4ビットのうちの1ビットを割り当てます．**図5**に回路図，**図6**にCポートの割り当てを示します．

8255Aには，Cポートに対してのみ，特定のビットを単独でON/OFFできる機能があるため，容易にSTROBE信号を出力することができます．

BUSY信号の状態は，Cポートを読み出すことでわかります．

図7にフローチャートを示します．参考までに8086でのプログラムを**リスト1**に示します．

〈図2〉(7) 8255Aの端子配置

Z80 PIO

Z80 PIOは，Z80ファミリの汎用8ビットの並列入出力インターフェース用のLSIです．このLSIは，次のような特徴をもっています．

- ▶40ピンDIPである
- ▶AとBの二つの入出力を任意に設定できる8ビット・ポートをもっている
- ▶各ポートは次の四つのモードに設定できる
  - モード0：出力モード
  - モード1：入力モード
  - モード2：双方向モード
  - モード3：ビット・モード
- ▶ハンドシェイクのための信号線を二つもっている
- ▶Z80のモード2割り込みのために，ベクトルの生成機能およびデイジィ・チェーンの割り込み処理機能を内蔵している
- ▶すべての入出力ラインは，TTLコンパチブルとなっている．またポートBは，ダーリントン・トランジスタなどの電流容量の大きい負荷を直接ドライブする能力をもっている．

Z80独自の機能をフルに利用しようとするとき，Z80ファミリの周辺インターフェース用のデバイスを

〈図4〉(7) 8255Aのコントロール・ワードの設定

ポート指定

	CS	A1	A0	WR	RD	
IN命令で，A, B, Cの各ポートのデータを読み取れる						入力動作（READ）
	0	0	0	1	0	ポートA→データ・バス
	0	0	1	1	0	ポートB→データ・バス
	0	1	0	1	0	ポートC→データ・バス
OUT命令で，A, B, Cの各ポートへデータを出力できる						出力動作（WRITE）
	0	0	0	0	1	データ・バス→ポートA
	0	0	1	0	1	データ・バス→ポートB
	0	1	0	0	1	データ・バス→ポートC
モード設定のためのコマンドを書き込む．このモード設定で各ポートの状態が決まる．$D_7=0$のときは，ポートCの各ビットのON/OFFの制御ができる						コントロール
	0	1	1	0	1	データ・バス→コントロール・レジスタ
						機能なし
	1	×	×	×	×	データ・バス→3ステート
	0	1	1	1	0	イリーガル状態
	0	×	×	1	1	データ・バス→3ステート

〈図5〉[3],[7] 8255A を使ったセントロニクス・インターフェース回路図

LS07 などのバッファを用いることができる．その場合，こちらのインターフェースでも5.6kでプルアップしておくほうがよい

受け(プリンタの)入力条件

一般的には約10kΩの抵抗でプルアップされたLS14などが入力になっている

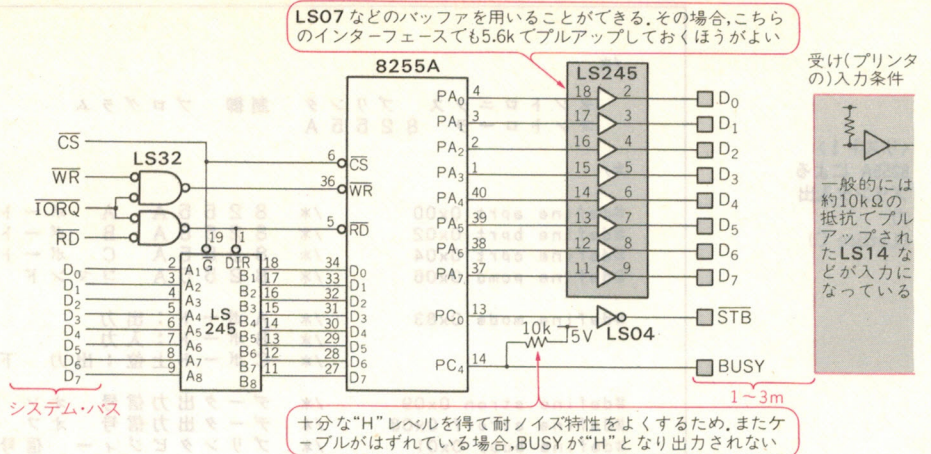
8255A　LS32　LS245　LS04

$\overline{CS}$　$\overline{WR}$　$\overline{IORQ}$　$\overline{RD}$

D_0～D_7　PA_0～PA_7　PC_0　PC_4　$\overline{STB}$　BUSY　10k　5V

システム・バス

1～3m

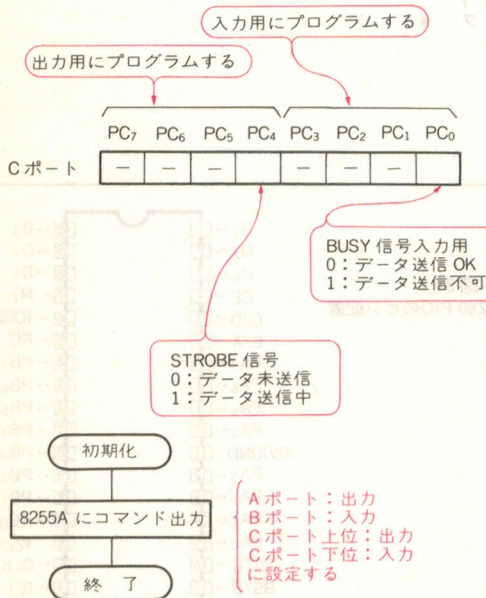
十分な"H"レベルを得て耐ノイズ特性をよくするため．またケーブルがはずれている場合，BUSYが"H"となり出力されない

(a)　入力回路

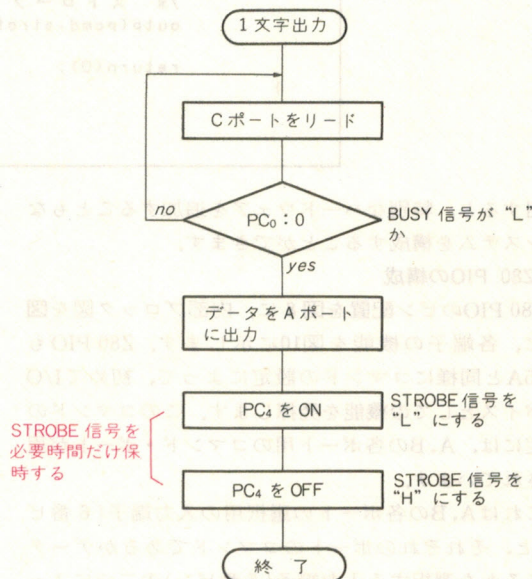
信　号　名	回　路　型　式
DATA 1~8	+5V, 1k, 7414 相当品
$\overline{DATA \cdot STB}$	+5V, 1k, 100p, 7414 相当品
$\overline{INPUT \cdot PRIME}$	+5V, 1k, 100p, 7414 相当品

(b)　出力回路

信　号　名	回　路　型　式
$\overline{ACK}$, $\overline{FAULT}$ BUSY, PE SELECT	+5V, 1k, 100p, 7406
+5V	+5V, 1k

(＊PC-PR406LP より)

〈図6〉 8255A Cポートの割り当て

〈図7〉 8255Aを使ったセントロニクス制御のフローチャート

(a)　8255A の初期化

(b)　1文字出力

〈リスト1〉
8255Aによるプリンタ入出力
(Lattice C)

```
/*
   セントロニクス  プリンタ  制御  プログラム
   コントローラ  8255A

*/

#define aprt 0x00        /* 8255A  A  ポート        */
#define bprt 0x02        /* 8255A  B  ポート        */
#define cprt 0x04        /* 8255A  C  ポート        */
#define pcmd 0x06        /* 8255A  コマンド  ポート  */

#define mode 0x83        /* Aポート：出力                  */
                         /* Bポート：入力                  */
                         /* Cポート上位：出力  下位：入力  */

#define stron 0x09       /* データ出力信号  オン  */
#define stroff 0x08      /* データ出力信号  オフ  */
#define busy 0x01        /* プリンタビジィー  信号  */

/*
   8255A  初期化  ルーチン
*/
pinit()
{
          outp(pcmd,mode); /*  モード  設定  */
          return(0);
}

/*
   プリンタ  1文字出力  ルーチン
*/
pout(data)
char data;
{
          int       i;

          /*  プリンタのビジィー  チェック  */
          while(inp(cprt)&busy){ }

          /*  1文字出力  */
          outp(aprt,data);
          /*  ストローブ  オン  */
          outp(pcmd,stron);
          for(i=0;i==2;i++){ }
          /*  ストローブ  オフ  */
          outp(pcmd,stroff);

          return(0);
}
```

使用すると，特別なハードウェアを追加することもなくシステムを構成することができます．

● Z80 PIOの構成

Z80 PIOのピン配置を図8に，内部ブロック図を図9に，各端子の機能を図10に示します．Z80 PIOも8255Aと同様にコマンドの設定によって，初めてI/Oデバイスとしての機能を発揮します．このコマンドの設定には，A,Bの各ポート用のコマンド・ポートが用意されています．

これはA,Bの各ポートの選択用の入力端子(6番ピン)と，それぞれのポートのコマンドであるかデータであるかを選択する入力端子(5番ピン)の二つによって，各ポートの選択が制御されます．

具体的には，これらの入力端子にアドレス・バスの

〈図8〉
Z80 PIOのピン配置

〈図9〉[7]
Z80 PIOの内部ブロック図

〈図10〉[7] Z80 PIOの機能説明

CPUデータ・バス：CPUのデータ・バスに接続され，PIOとZ80CPUとのデータ・コマンドの受け渡しはすべてこのバス経由で行われる

D_0–D_7（ピン19, 20, 1, 40, 39, 38, 3, 2）

PIO制御線
- ポートB, Aの選択．選択用信号はCPUからのA_1　"H"→ポートB,"L"→ポートA（B/A SEL, 6）
- 制御信号，データの選択．選択信号－A_0　"H"→コマンド,"L"→データ転送（C/D SEL, 5）
- チップ・イネーブル（アクティブ"L"）（CE, 4）
- 書き込みサイクル　CPU→PIO
- 読み出しサイクル　PIO→CPU

M1（37），IORQ，RD（35）

割り込み制御線
- 割り込み要求（オープン・ドレイン，アクティブ"L"）（INT, 23）
- 割り込みイネーブル入力（アクティブ"H"）（IEI, 24）
- 割り込みイネーブル出力（アクティブ"H"）（IEO, 22）

デイジィ・チェーン形成のため用いる

φ（25），+5V（26），GND（11）

ポートA入出力：A_0–A_7（ピン15, 14, 13, 12, 10, 9, 8, 7）

ハンドシェイク
- ARDY（18）：レジスタAレディ（アクティブ"H"）
- ASTB（16）：周辺装置が与えるポートAストローブ・パルス（アクティブ"L"）

ポートB入出力：B_0–B_7（ピン27, 28, 29, 30, 31, 32, 33, 34）

ハンドシェイク
- BRDY（21）：レジスタBレディ（アクティブ"H"）
- BSTB（17）：周辺装置が与えるポートBストローブ・パルス（アクティブ"L"）

M1の役割

M1・IORQ　共に"L"の時，割り込みベクトルをCPUに出す

M1・RD　共に"L"の時は，CPUの命令コード読み出しサイクルなので，Z80 PIOは何もしない

上記以外の時，M1が"L"になると，PIOはリセットされる

PIOの制御

CE, RD, IORQ 共にアクティブ → B/A選択で選択されたポートからCPUへデータを転送（読み込み動作）

CE, IORQ 共にアクティブ，RD 非アクティブ → B/Aで選択されたポートへCPUからC/D選択で指定された情報（データまたはコマンド）が送られる

CPUが割り込み受け付け状態 → M1, IORQ共にアクティブになり，割り込みを出しているポートから自動的に，CPUデータ・バス上へポートの割り込みベクトルが送られる

PIOのリセット

M1端子（37）に，システム・バスのM1と，RESET（システム・バスのRESETを反転させる）のORを加えると，RESETで，PIOも同時にリセットすることができる．

RESETは，通常マシン・サイクルに対して十分長い期間"L"になる

ハンドシェイク信号は，モードによって意味が異なる

RDY（READY）
- **出力モード**：周辺装置へのデータ転送が準備できたことを示すため，アクティブとなる
- **入力モード**：入力レジスタが空となり，周辺装置からのデータ受け入れ準備ができた時アクティブとなる
- **双方向モード**：ARDY-ポートAの出力レジスタ内容の出力準備が完了した時アクティブとなる．BRDY-ポートAのデータ受け入れ準備ができた時アクティブとなる
- **ビット制御モード**：強制的に"L"状態になる

STB（STROBE）
- **出力モード**：周辺装置がPIOからデータを受信したことを通知する信号．立ち上がりが有効
- **入力モード**：周辺装置から，入力レジスタへデータをロードした時に出される．アクティブになった時，データがPIOにロードされる
- **双方向モード**：ASTBがアクティブの時，ポートAの出力レジスタからのデータが，ポートAの双方向データ・バス上にのせられる．信号が立ち上がれば周辺装置がデータを受け取ったとみなされる．BSTBは，周辺装置からポートAの入力レジスタへの データのストローブ（書き込み）に用いられる
- **ビット制御モード**：無効

A_0, A_1を接続すると，表1に示すようにそれぞれに対応したポート・アドレスが設定できます．

8255Aでもハンドシェイクの必要な場合は，A, Bの2ポートしか使用できません．その場合，割り込み処理の機能を内蔵したZ80 PIOのほうが有利になります．

◆ Z80 PIOのモードの説明

● PIOのモード0

モード0は，バイト・データをデータ・ポートに出力します．図11にそのタイミングを示します．

CPUからのOUT命令によってデータの出力を開始します．PIOが出力要求を受け取ると，データ・バス上のデータをデータ・レジスタにラッチしてデータ・ポート上に出力します．WRが立ち上がると，クロックの立ち下がりのタイミングで，READYを“H”にします．これでデータの出力準備ができました．

データを受け取った装置は，STROBE信号を“L”にして，その立ち上がりのタイミングでPIOはREADYを“L”にして割り込みを発生させます．

● PIOのモード1

モード1は，バイト・データを装置から入力します．図12にそのタイミングを示します．

PIOにデータを送りたい装置は，STROBE信号を“L”にすることによりPIOを起動させます．STROBEが“L”になるとデータ・ポートのデータを入力データ・レジスタにラッチします．この信号の立ち上がりでREADY信号を“L”にして，割り込みを発生させデータがあることを知らせます．

CPUがIN命令を出すと，データ・レジスタのデータをデータ・バス上に出力して，RD信号の立ち上がりでREADYを“H”にします．

● PIOのモード2

モード2は，モード0とモード1の組み合わせで，各2組のREADYとSTROBE信号を使ってハンドシェイクを行います．そのために，ポートAだけが双方向となり，ポートBは，ビット・モードでしか使用できません．

このときポートBの割り込みは，ハンドシェイクに

〈表1〉[7] Z80 PIOとA_0, A_1の接続例

	A_1 C/D	A_0 A/B		A_1 A/B	A_0 C/D
Aポート・データ	0	0	Aポート・データ	0	0
Bポート・データ	0	1	Aポート・コマンド	0	1
Aポート・コマンド	1	0	Bポート・データ	1	0
Bポート・コマンド	1	1	Bポート・コマンド	1	1
データ，コマンド・ポートが連続してアドレスとなる			それぞれポートのデータ，コマンドが連続したアドレスとなる		

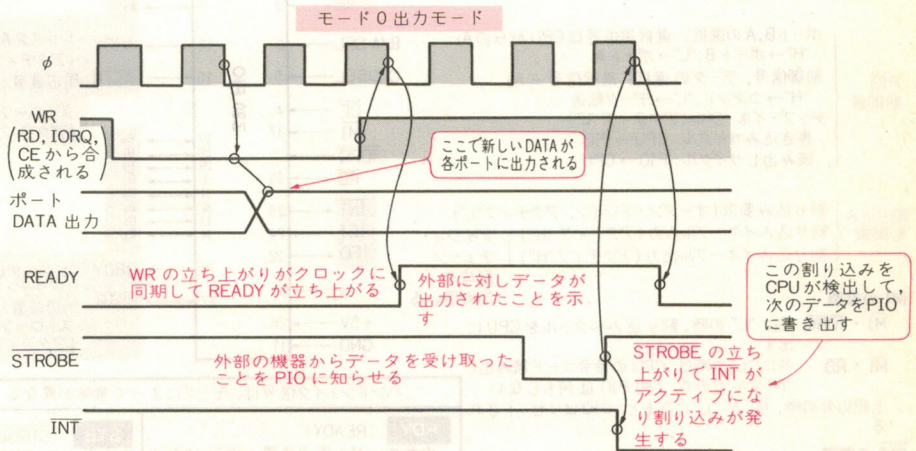
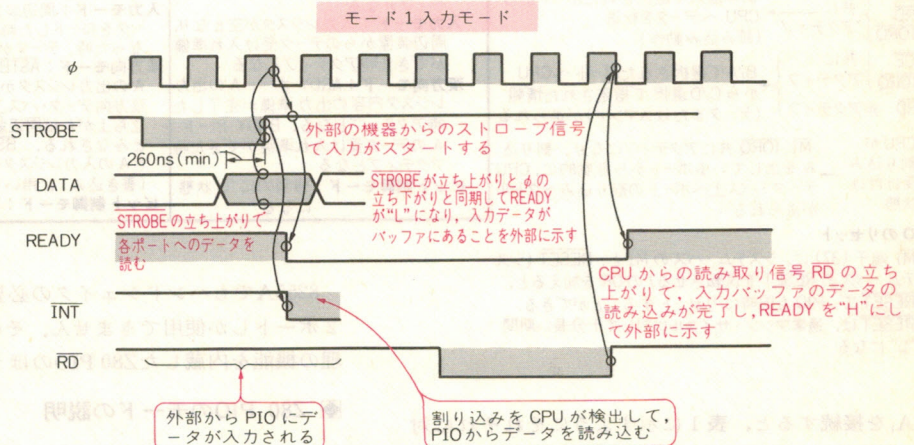
〈図11〉[7] Z80 PIOモード0のタイミング図

〈図12〉[7] Z80 PIOのモード1のタイミング図

使用されるために，割り込みを使わないポーリングで確認する必要があります．

図13にモード2のタイミングを示します．出力のタイミングは，モード0と同様であり，入力のタイミングは，モード1と同じです．

● PIOのモード3

モード3はビット・モードと呼ばれ，ハンドシェイクなしに，直接ポート上のデータを入出力します(図14)．

◈ Z80 PIOのプログラミング

● Z80 PIOの初期設定は数ステップのコマンド設定を必要とする

Z80 PIOの初期設定はA, Bの各ポートに対して，必要に応じて三種類のコマンドを書き込むことで行います．その三種類のコマンドは，次に示すものです(図15)．

(1) 割り込みベクトル

D_0ビットがゼロのコマンドは，割り込みベクトルとみなされます．これはZ80の割り込みを，モード2で実行する場合に必要になります．

(2) モード設定

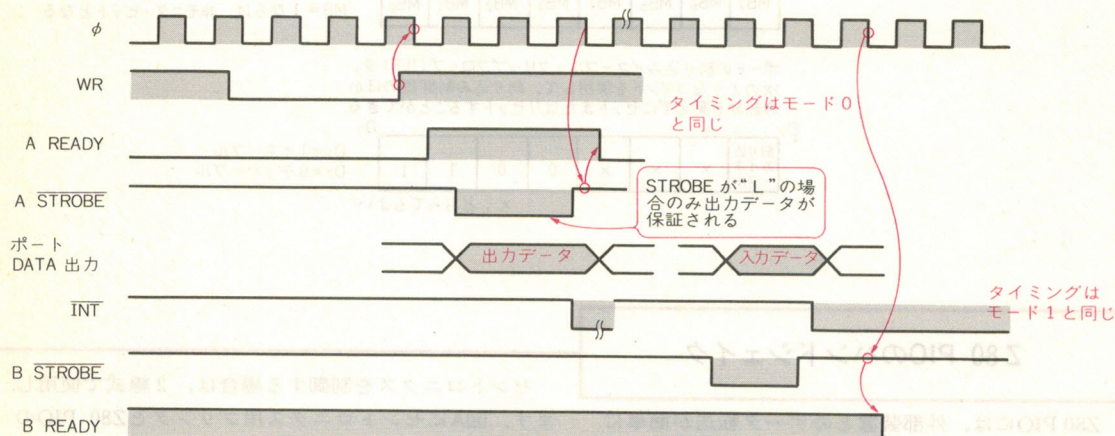
Z80 PIOで設定可能な四つのモードを指定するためのコマンドです．D_7, D_6ビットの組み合わせで，図に示すように0から3までのモードが決まります．このコマンドはD_0からD_3の4ビットがともに1となっています(図16)．

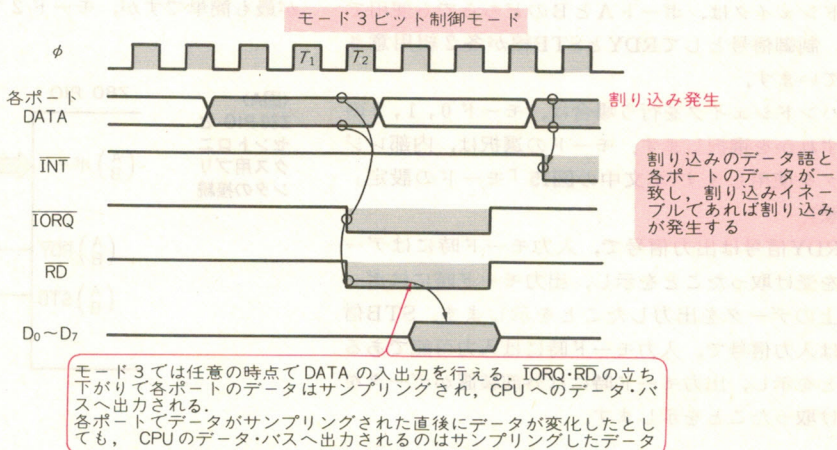
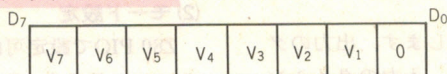
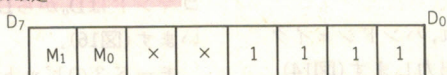
モード3のビット制御モードを指定した場合は，次に各ビットの入出力を決めるためのコマンドを書き込みます．各ビットは1で入力，0で出力になります．

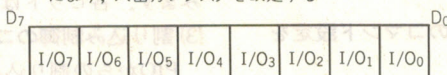
(3) 割り込み制御のコマンド

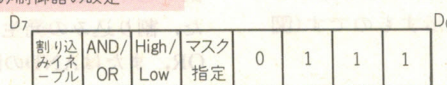
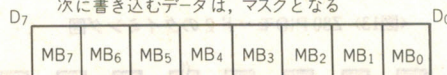
PIOからの割り込み要求の可否を制御するコマンドです．ビット制御モードに対しては，各ビットごとに割り込みの必要の有無を指定することができます．また，割り込みの発生するための条件を，ビット同士のOR，またはANDの関係からも指定することができます．

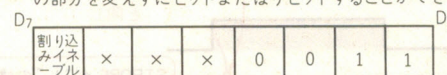
このコマンドは，図15に示すように下位4ビットが7Hとなっていて，ビット制御モード以外では，割り

〈図13〉 Z80 PIOモード2のタイミング図

〈図14〉[7]
Z80 PIOのモード3のタイミング図

〈図15〉(7)
Z80 PIOの制御語

M_1	M_0	モード
0	0	モード0　出力
0	1	モード1　入力
1	0	モード2　双方向
1	1	モード3　ビット制御

モード3を指定した場合，次に書き込むデータにより，入出力レジスタを設定する

D_7							D_0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O＝1ならば，入力ビットとなる
I/O＝0ならば，出力ビットとなる

3. 割り込み制御語の設定

マスク指定(Mask follows)が"H"レベルならば，次に書き込むデータは，マスクとなる

MB＝0ならば，モニタ・ビットとなる
MB＝1ならば，非モニタ・ビットとなる

ポートの割り込みイネーブル・フリップフロップ(IFF)を，次のようなコマンドを使用して，割り込み制御語のほかの部分を変えずにセットまたはリセットすることができる

D_7＝1イネーブル
D_7＝0ディセーブル

×：どちらでもよい

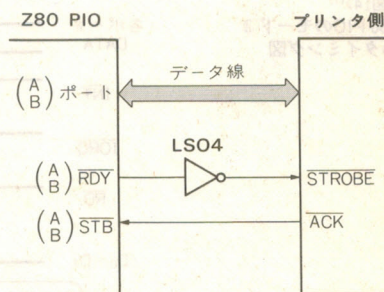
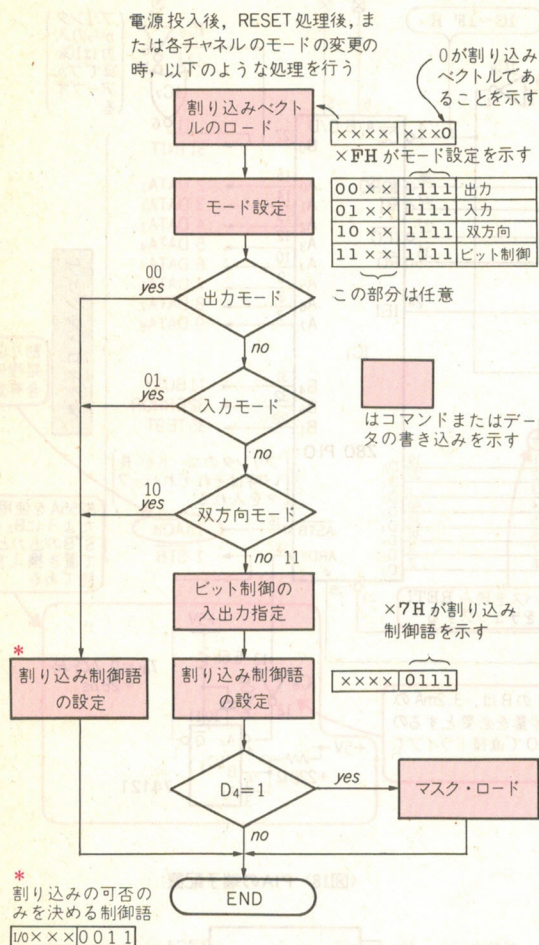
Z 80 PIOのハンドシェイク

Z80 PIOには，外部装置とのデータ転送が簡単に行えるように，ハンドシェイク機能があります．ハンドシェイクは，ポートAとBのどちらでも使用でき，制御信号としてRDYとSTB線が各2組用意されています．

ハンドシェイクを行う場合は，モード0，1，2のいずれかを選択します．モードの選択は，内部レジスタを使用します(本文中の図15「モードの設定」参照)．

RDY信号は出力信号で，入力モード時にはデータを受け取ったことを示し，出力モード時にはポート上のデータを出力したことを示します．STB信号は入力信号で，入力モード時には入力可能であることを示し，出力モード時には外部装置がデータを受け取ったことを示します．

セントロニクスを制御する場合は，2線式で使用します．図Aにセントロニクス用プリンタとZ80 PIOの接続を，図Bに制御タイミングを示します．この方法が最も簡単ですが，モード2では使用できません．

〈図A〉
Z80 PIOとセントロニクス用プリンタの接続

〈図16〉[7] Z80 PIOのコマンド設定のフローチャート

込み発生の有無の制御のみを行います．

◆ セントロニクスへの応用

Z80 PIOの具体的な使用例として，セントロニクス・インターフェースの回路を考えます．このインターフェースでは，Z80の割り込み機能を利用できるようにしてあります(図17)．

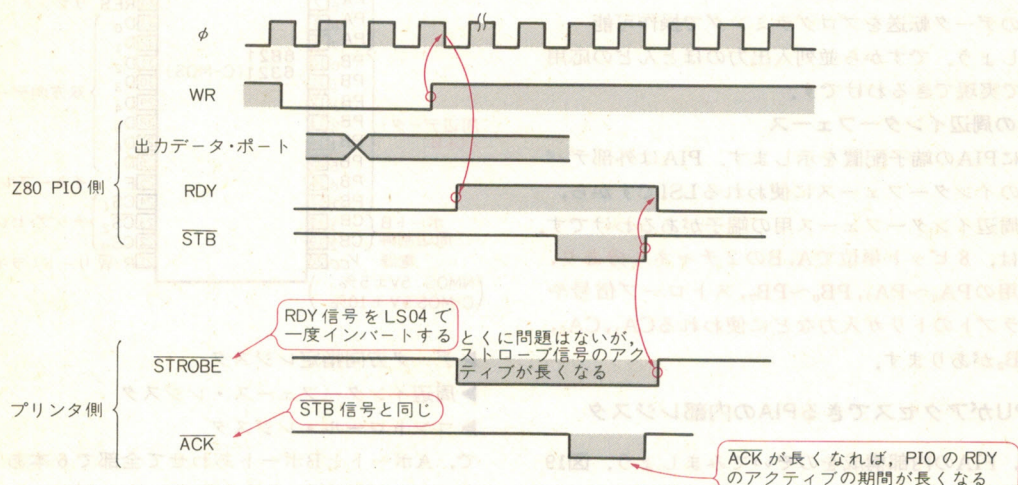
この回路のためのプログラムを，**リスト2, リスト3**に示します．アセンブラとターボ・パスカルのプログラムを示してあります．

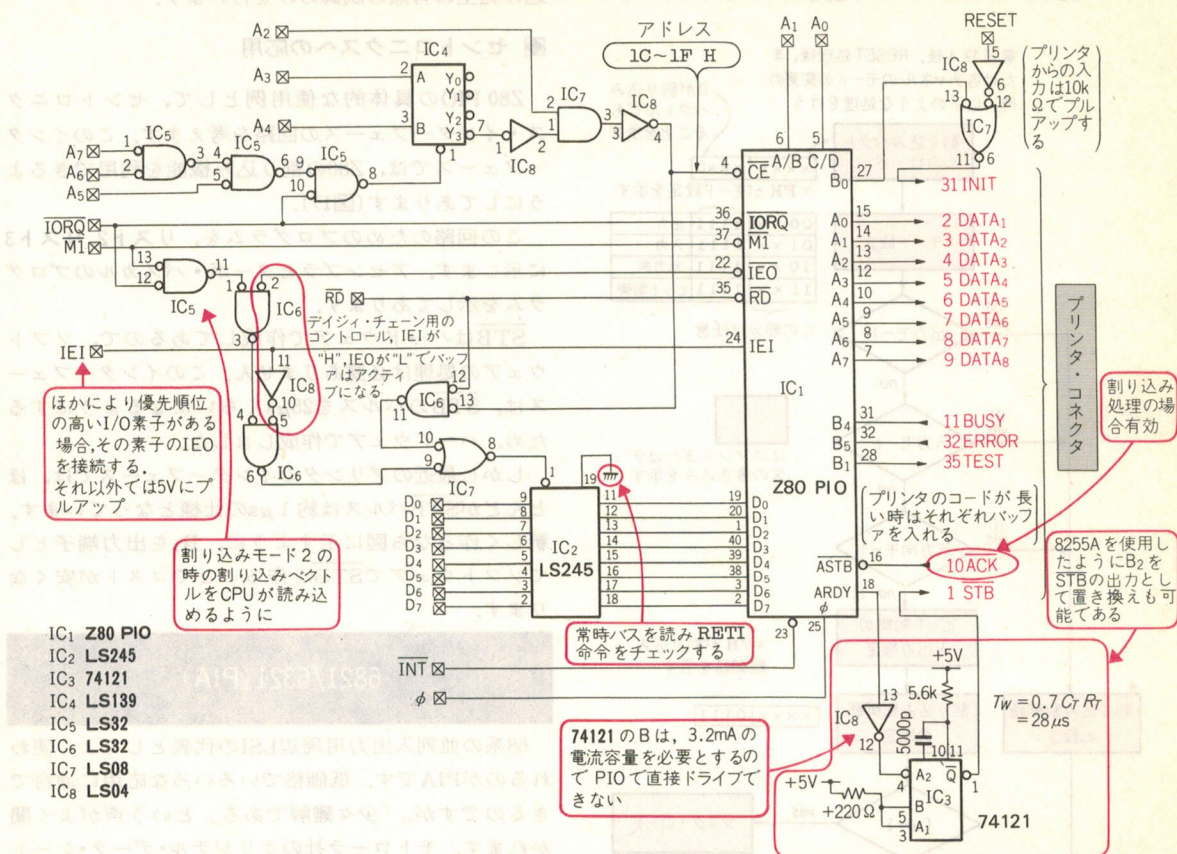
$\overline{\text{STB}}$はハードウェアで作成してあるので，ソフトウェアの処理は必要ありません．このインターフェースは，$\overline{\text{STB}}$のパルスを25μsくらいになるようにするため，ハードウェアで作成しました．

しかし最近のプリンタのインターフェースでは，ほとんどが$\overline{\text{STB}}$パルスは約1μsの仕様となっています．新しく作るなら図に示すように，B_2を出力端子としてソフトウェアで$\overline{\text{STB}}$を作ることでコストが安くなります．

6821/6321(PIA)

68系の並列入出力用周辺LSIの代表としてよく使われるのがPIAです．低価格でいろいろな応用に適合できるのですが，「少々難解である」という声がよく聞かれます．モトローラ社のオリジナル・データ・シートでは機能と諸特性だけだったのが，日立製のマニュアルではたんなる直訳ではなく，内部の動作原理まで懇切丁寧に独自の書き方で解説，記述してあり，初めて

〈図B〉 図Aの接続によるPIOのタイミング

〈図17〉 Z80 PIOを用いたプリンタ・インターフェース

IC₁ **Z80 PIO**
IC₂ **LS245**
IC₃ **74121**
IC₄ **LS139**
IC₅ **LS32**
IC₆ **LS32**
IC₇ **LS08**
IC₈ **LS04**

使うときには参考になります．

● **PIAのもつ機能**

PIAの機能を列挙してみると，

▶たんなるたれ流し式のラッチI/O
▶ストローブ付きのI/O
▶ハンドシェイクによる転送
▶上記のデータ転送をプログラミングで操作可能

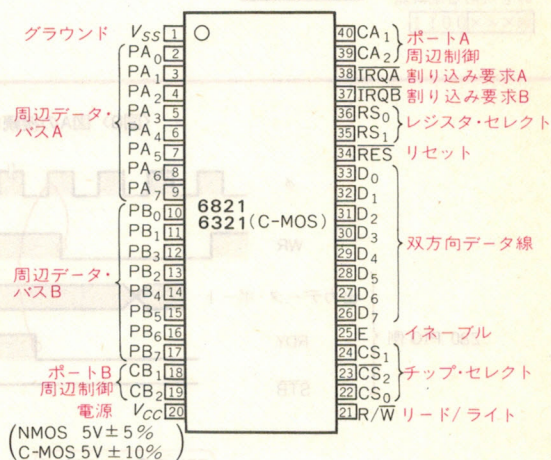
などでしょう．ですから並列入出力のほとんどの応用はPIAで実現できるわけです．

● **PIAの周辺インターフェース**

図18にPIAの端子配置を示します．PIAは外部デバイスとのインターフェースに使われるLSIですから，当然，周辺インターフェース用の端子があるわけです．これらは，8ビット単位でA，Bの2チャネル分あり，データ用のPA_0～PA_7，PB_0～PB_7，ストローブ信号やインタラプトのトリガ入力などに使われるCA_1，CA_2，CB_1，CB_2があります．

◈ MPUがアクセスできるPIAの内部レジスタ

さて，PIAの内部構成をのぞいてみましょう．図19にPIAの内部ブロック図を示します．MPUがアクセス可能なPIAのレジスタは，

〈図18〉 PIAの端子配置

▶データ方向指定レジスタ
▶周辺インターフェース・レジスタ
▶コントロール・レジスタ

で，AポートとBポートあわせて全部で6本あります．

● **DDRA(B)(データ方向指定レジスタ)**

データ方向指定レジスタは，I/Oデータの方向を決

〈リスト2〉[7] Z80 PIOを用いたアセンブラによるプリンタ制御例

```
                              ; program Z80502.MAC
                              ; Z80-PIO test routine
                              ; 85/02/10 Y.Kanzaki
                              ;
001D                          piopac   equ  01Dh        ; PIO port address      ┐Z80 PIOの各ポート
001C                          piopad   equ  01Ch        ;                       │のアドレス
001F                          piopbc   equ  01Fh        ; PIO port address      │
001E                          piopbd   equ  01Eh        ;                       ┘
000F   Z80 PIOモード0の設定コマンド     cmd_out  equ  00Fh        ; PIO
00CF   Z80 PIOモード3の設定コマンド     cmd_bit  equ  0CFh        ; PIO
00F0   ビット制御モードでのI/O設定     cmd_io   equ  0F0h        ; in D7-D4 out D3-D0
                              ;
                                       org
0000'   3E 0F   Z80 PIOのモード   pioinit:ld   A,cmd_out     ;┐ポートAをモード0に設定
0002'   D3 1D   設定ルーチン              out  (piopac),A    ;┘
0004'   3E CF                           ld   A,cmd_bit     ;┐ポートBをビット制御モー
0006'   D3 1F                           out  (piopbc),A    ;│ドにして,各ビットの入出
0008'   3E F0                           ld   A,cmd_io      ;┘力を設定
000A'   D3 1F                           out  (piopbc),A    ;
000C'   C9                              ret
                              ;
000D'   DB 1E   プリンタの状況の  lstat:  in A,(piopbd)       ; プリンタが受信可能でな
000F'   CB 67   チェック                 bit 4,A             ; ければA=OFFHとして
0011'   3E FF                           ld A,0ffH           ; もどる.
0013'   C8                              ret Z               ; プリンタが受信可能なら
0014'   AF                              xor A               ; A=O
0015'   C9      プリンタへの出力          ret                 ;
                ルーチン
                              ;
0016'   CD 000D' STBはZ80 PIO よ list:  call lstat          ;┐プリンタが受信可能にな
0019'   B7      り出力される             or A                ;┘るまでループする
001A'   28 FA                           jr z,list           ;
001C'   79                              ld A,C              ;┐Cレジスタにセットされた
001D'   D3 1C                           out (piopad),A      ;┘データを,プリンタに出力
001F'   C9                              ret                 ; する
                              ;
                                        end                 ;
```

定するレジスタで，各ポート，各ビットごとに指定できます．図20のようにAポートのビット3，4，7を出力に，そのほかは入力に割り付けるというような使い方ができるわけです．

なぜ，DDRA(B)が最初に登場するのかというと，リセット直後にPIAの内部アドレス0または内部アドレス2を読み込むと，このレジスタの内容が得られます．また，書き込み動作を行っても同様にDDRA(B)に書き込まれます．すなわち，PIAの最も基本的な項目であるデータ方向は，初期化のときに決まっていなければならないからです．

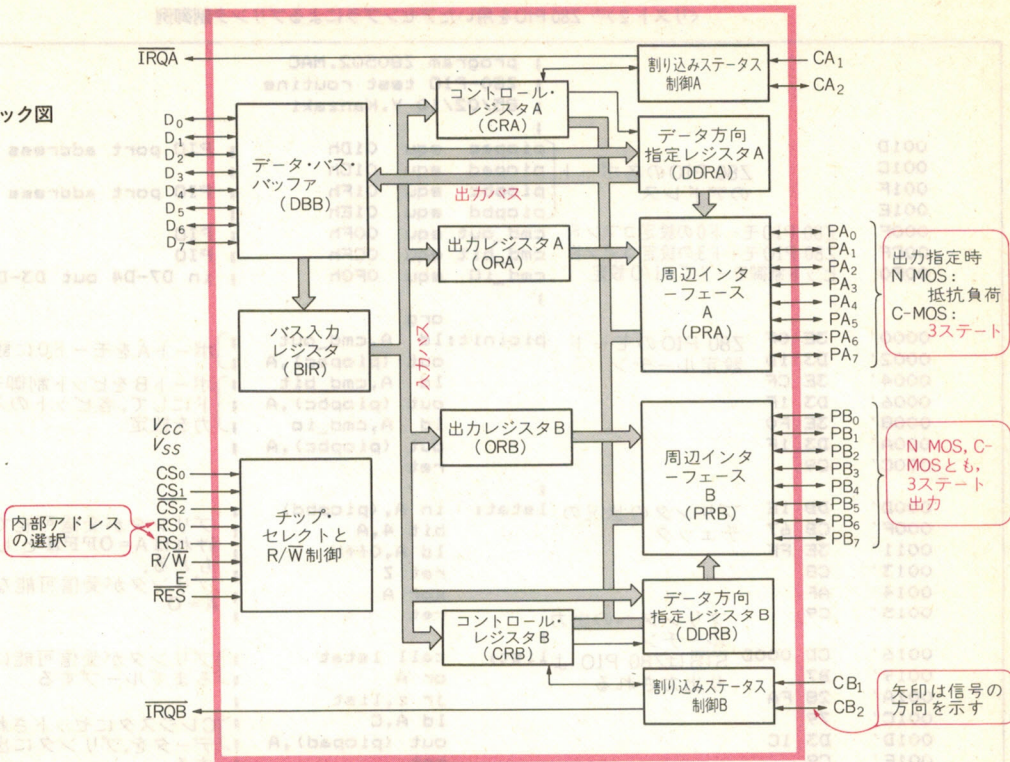
● PRA(B)(周辺インターフェース・レジスタ)

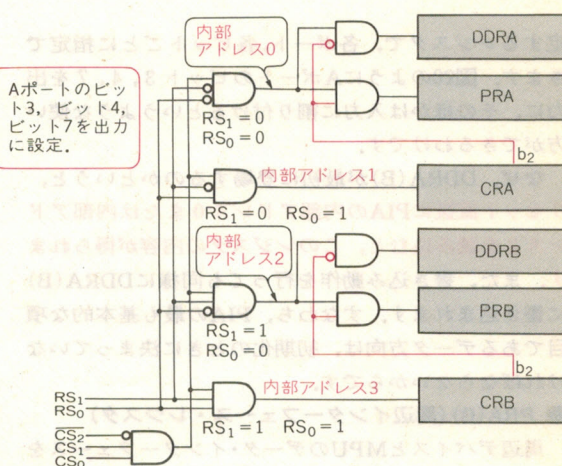
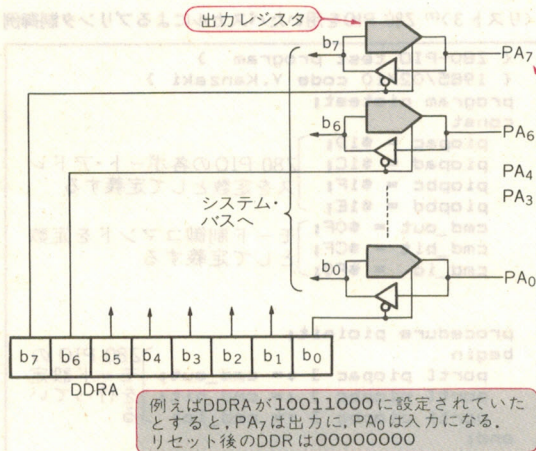
周辺デバイスとMPUのデータ・インターフェースを行うPRA(B)とDDRA(B)は内部アドレスが同じですが，コントロール・レジスタCRA(B)のビット2の値によって切り替えることができます．すなわち，CRA(B)のビット2が0のとき，内部アドレス0，2はDDRA(B)が選択されていますが，ここを“1”にしてやるとPRA(B)が現れるというわけです．図21にPIAレジスタ選択の概念図を示します．

このレジスタは，I/Oデータそのものを取り扱うもので，MPUがPRA(B)をリード/ライトすることが，すなわち周辺デバイスとのデータのやりとりになります．

〈リスト3〉[7] Z80 PIOを用いたパスカルによるプリンタ制御例

```
{ Z80-PIO test program  }
{ 1985/02/10 code Y.Kanzaki }
program piotest;
const
  piopac  = $1D;  ┐Z80 PIOの各ポート・アドレ
  piopad  = $1C;  │スを定数として定義する
  piopbc  = $1F;  │
  piopbd  = $1E;  ┘
  cmd_out = $0F;  ┐モード制御コマンドを定数
  cmd_bit = $CF;  │として定義する
  cmd_io  = $F0;  ┘

procedure pioinit;
begin                                   ┐Z80 PIOの
  port[ piopac ] := cmd_out;            │モード設定
  port[ piopbc ] := cmd_bit;            │を行ってい
  port[ piopbc ] := cmd_io;             ┘る
end;

function lstat:boolean;     プリンタの状態
begin                       チェックの関数
 lstat := (port[ piopbd ] and $10)= 0;
end;

procedure list( data : byte );
begin
 repeat          ┐プリンタが受信可になるまでまっ
 until lstat;    │てI/Oポートpiopadへ出力データ
 port[ piopad ] := data;  ┘data を書き出す
end;

begin
  pioinit;          ┐テストのために一つのCR
  list( $31 );      │を出力するプログラム
  list( $0D );      ┘
end.
```

〈図19〉
PIAの内部ブロック図

〈図20〉 DDR(データ方向指定レジスタ)の概念

〈図21〉 PIAのレジスタの選択

● CRA(B)(コントロール・レジスタ)

MPUと周辺デバイスとのデータ転送をする場合，両者間の処理能力の関係や突発的なデータの送信により，正しいデータを見逃す可能性があります．したがって，PRA(B)のたんなるリード/ライトでは信頼性のあるデータの転送はできません．

ですから，ストローブやハンドシェイクという手法を使うことになります．CA(B)$_{1,2}$ 端子はこのために使われるもので周辺制御線とも呼ばれます．コントロール・レジスタの役割はこれらの周辺制御線をコントロールすることと，前述したDDRA(B)とPRA(B)の切り替えおよび割り込み信号の処理です．

PIAが難解といわれるのは，この周辺制御線をどのようにうまく目的のデータ転送に応用するかといった点でしょう．そのためにCA(B)$_{1,2}$について詳しく考察してみます．

CA(B)$_1$は入力専用で，外部事象の変化を捕らえることができます．CA(B)$_2$は入出力どちらにも設定でき，入力に設定した場合にはCA(B)$_1$と同じような動作をします．これらの設定は，もちろんコントロー

〈図22〉
CA(B)$_1$関連のコントロール・レジスタ

PIA

CA(B)$_1$

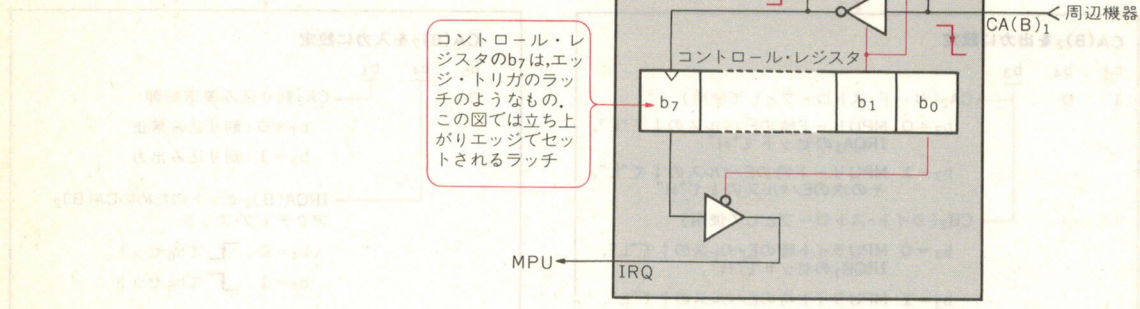
周辺機器

コントロール・レジスタのb$_7$は,エッジ・トリガのラッチのようなもの.この図では立ち上がりエッジでセットされるラッチ

コントロール・レジスタ

b$_7$ b$_1$ b$_0$

MPU

IRQ

ル・レジスタによって行います.

◈ コントロール・レジスタの働き

図22にCA(B)$_1$関連のコントロール・レジスタの働きを示します.関連ビットの意味は,

▶ビット7は,CA(B)$_1$の変化(ビット1で設定)があったことを示すフラグで読み出し専用です.

▶ビット0は,ビット7の内容を割り込み信号として,ハード的にMPUへ出力することを許すフラグです.すなわち,CA(B)$_1$に変化があったときのインタラプト・イネーブル/ディセーブルを決めます.

▶ビット1はCA(B)$_1$の変化の方向を決めるもので,周辺デバイスのストローブ信号を↑エッジでも↓エッジでも捕らえることができますから,周辺デバイスの仕様に合わせることができます.

なお,CA(B)$_2$を入力にセットするときは,ビット

〈図23〉 コントロール・レジスタのビット割り付けとCA(B)端子の設定

CA(B)$_2$を出力に設定

b_5 b_4 b_3
1 0

→ CA_2(リード・ストローブとして使用)
- b_3＝0 MPUリード時のEパルスの↓で“L”, $IRQA_1$のセットで“H”
- b_3＝1 MPUリード時のEパルスの↓で“L”, その次のEパルスの↓で“H”

→ CB_2(ライト・ストローブとして使用)
- b_3＝0 MPUライト時のEパルスの↑で“L”, $IRQB_1$のセットで“H”
- b_3＝1 MPUライト時のEパルスの↑で“L”, その次のEパルスの↑で“H”

b_5 b_4 b_3
1 1

→ CA_2のセット/リセット
- b_3＝0 CA_2は“L”
- b_3＝1 CA_2は“H”

CA(B)$_2$を入力に設定

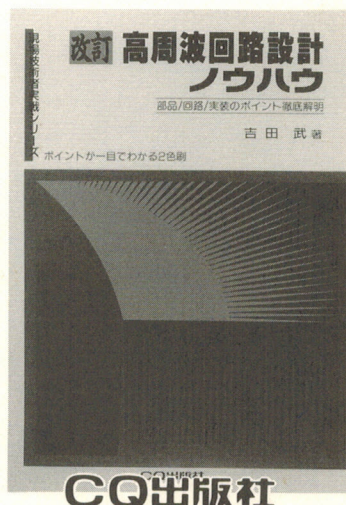
b_5 b_4 b_3
0

→ CA_2割り込み要求制御
- b_3＝0：割り込み禁止
- b_3＝1：割り込み出力

→ IRQA(B)$_2$セットのためのCA(B)$_2$アクティブ・エッジ
- b_4＝0：⏋_ でb_6セット
- b_4＝1：_⌈ でb_6セット

5を“0”にします．こうすることにより，CA(B)$_2$もCA(B)$_1$と同じ機能をもつことになります．このとき，ビット3がビット0と，ビット4がビット1と同様の意味をもちます．

● CA_2とCB_2は違う

図23にCA(B)の制御のしかたをまとめて示します．ここまでの解説とPIAの内部ブロック図からは，CA_2とCB_2の違いはわかりません．しかし，この二つの信号線には，大きな違いがあるのです．図23をみてください．CA_2, CB_2を出力に設定したとき，

▶CA_2はAポートからデータを読み出すことによってセットされる．

▶CB_2はBポートにデータを書き込むことによってセットされる．

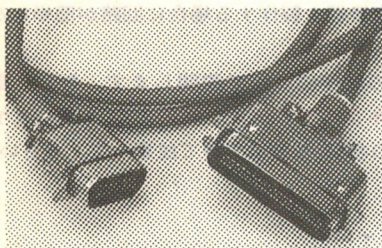
ということになります．この意味を考えると，周辺デバイスに対して，

▶CA_2が「データを受け取った」

▶CB_2が「データを送った」

の確認信号を出していると取れるわけです．したがって，PIAをハンドシェイクで使う場合，Aポートは入力に，Bポートは出力に使うと便利であることがわかります．

§ 1-3 セントロニクス↔RS-232C 変換器の製作

土屋 哲/矢吹貞人

セントロニクス・インターフェースとRS-232Cインターフェースは，パソコンの標準インターフェースといってもよいくらい，ほとんどのパソコンで最低どちらか一種類が標準装備されてきています．

また，パソコンと接続する周辺装置も，セントロニクス・インターフェース,RS-232Cインターフェースを備えたものが多くなっています．

そこで，セントロニクス→RS-232C変換器やRS-232C→セントロニクス変換器があれば，パソコンと周辺装置およびほかのパソコンとの接続の可能性が増大することになり，パソコン間でプログラムやデータの転送を行ったり，機器の有効利用が図れると考え，セントロニクス↔RS-232C変換器を製作しました．

回路構成

本器のブロック図を図1に示します．本器はUART(Universal Asynchronous Receiver Transmitter)を中心にして，大きく三つの部分に分けられます．

一つは，ボーレート・ジェネレータです．UARTの下側の部分で，UARTに必要とするボーレートに対応したクロックを供給します．

もう一つは，RS-232C→セントロニクス変換部です．UARTおよびUARTの左側の部分で，RS-232Cのシリアル・データをセントロニクスのパラレル・データに変換します．

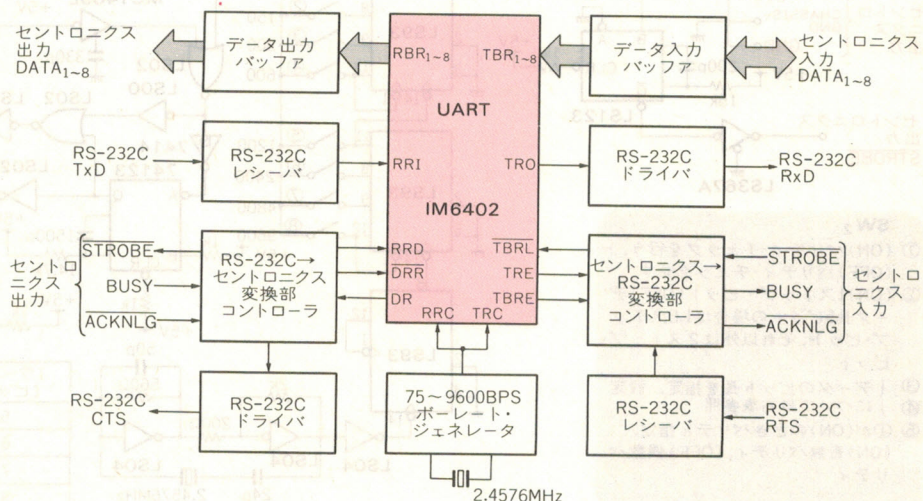
残りの一つはセントロニクス→RS-232C変換部です．UARTおよびUARTの右側の部分で，セントロニクスのパラレル・データを，RS-232Cのシリアル・データに変換します．

本器の回路図を図2に示します．次に各部分について説明します．

● ボーレート・ジェネレータ

ボーレート・ジェネレータ部は，2.4576MHzの水晶とLS04による発振回路の出力を，LS93を3個用いて分周する回路です．ボーレート 75, 150, 300, 600, 1200, 2400, 4800, 9600のそれぞれ16倍のクロック(UART，インターシル社のIM6402は，ボーレートの16倍のクロックを必要とする)として取り出しています．

本器のクロックは，受信部,送信部とも同一の周波数を用いていますが，受信部,送信部別の周波数(すなわち別のボーレート)で動作させることも可能です．ボーレートの切り替えは，SW_1(ボード上のDIP SW)により行います．

〈図1〉セントロニクス↔RS-232C変換器のブロック図

● RS-232C→セントロニクス変換部

RS-232C→セントロニクス変換部は，図1，図2のUARTおよびUARTの左側の部分です．

本器で使用したUARTは，インターシル社のC-MOS LSIのIM6402で，消費電力は10mW以下と低消費電力です．そのうえワード長の設定，パリティ・チェック有無の設定，奇数か偶数かのパリティ選定，ストップ・ビット長の設定がそれぞれのピンの電圧レベルを設定するだけで行えます．本器のようにハードウェアのみで変換器を製作する場合に回路が簡単になり有利です．

次に回路の動作について説明します．なお本器のRS-232Cコネクタは，モデム定義となっていますので，各信号の名前，役割について注意してください．

〈図2〉全体の回路図

SW₂
① (ON)パリティ・チェックを行う，(OFF)パリティ・チェックせず
② (ON)1ストップ・ビット(OFF)データ長5ビットの場合は1.5ストップ・ビット，それ以外は2ストップ・ビット
③ ④ }データのビット長を指定，設定については右表参照
⑤ ①が(ON)のときパリティ指定，(ON)奇数パリティ，(OFF)偶数パリティ

1ワード長(ビット)	SW₂ ③	SW₂ ④
5	(ON)	(ON)
6	(ON)	(OFF)
7	(OFF)	(ON)
8	(OFF)	(OFF)

① まず，RS-232Cの送信データ(TxD)はRS-232CレシーバMC1489LでTTLレベルに変換され，UARTの受信部のRRI端子に加えられます．

② 次に，UARTの受信部はRRI端子に加えられたシリアル・データのスタート，データ，パリティとストップ・ビットを受け，パリティとストップ・ビットが正しいことを確認した後，パラレル・データに変換します．

③ このとき，UARTのDR端子は"H"レベルに変わりますので，この信号を反転し，UARTの$\overline{\mathrm{RRD}}$端子に加え，UARTのパラレル・データ(RBR_1～RBR_8)を読み出し，LS367Aによるバッファを通してセントロニクス出力の$DATA_1$～$DATA_8$とします．

④ 一方，DRが"L"から"H"に変わったときから約1μs遅れた約1μs幅のパルスが，LS123による単安定マルチバイブレータ2段の回路により作られ，セントロニクス出力の$\overline{\mathrm{STROBE}}$となります．

⑤ セントロニクス側の動作が完了し，セントロニクス出力の$\overline{\mathrm{ACKNLG}}$が"H"から"L"に変わったとき，LS123の単安定マルチバイブレータが約0.5μsのパルスを出します．そして，これをUARTの$\overline{\mathrm{DRR}}$に加えDRを"L"レベルにし，UARTの受信部が次のシリアル・データを受信できるようにします．

⑥ 一方，DRとセントロニクス出力のBUSY信号のORをとり，RS-232CドライバSN75150Pを通してRS-232Cの送信可(CTS)に出しています．

このRS-232C→セントロニクス変換部のタイムチャートを図3に示します．本器では，セントロニクス関係のタイミングは，エプソン社のプリンタTP80のタイミングを参考にして設計してあります．

シリアル信号のデータ長，パリティ・チェックの有無，ストップ・ビット長の設定は，セントロニクス→RS-232C変換部と共通で，SW_2(ボード上のDIP SW)で行います．SWの設定法については，図2を参照してください．

RS-232Cコネクタは，セントロニクス→RS-232C変換部と共通です．RS-232Cコネクタの信号配置を表1に示します．本変換部では，動作に必要な信号は送信データ(TxD)，送信可(CTS)および信号接地(SG)のみですが，相手方の動作のためにデータ・セット・レディ(DSR)，キャリア検出(CD)の各々は，＋12Vにプルアップしてあります．

セントロニクス出力コネクタの信号配置を表2に示します．

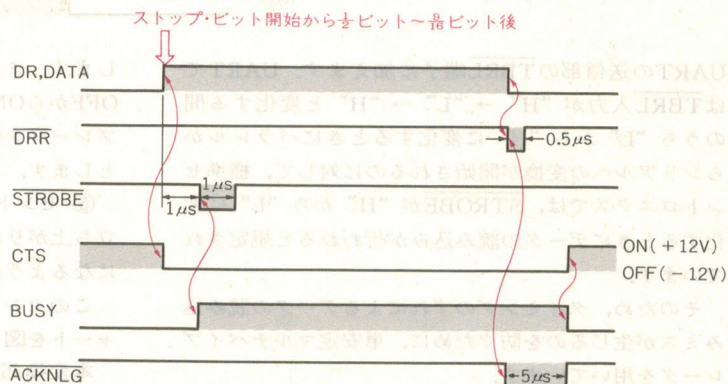
● セントロニクス→RS-232C変換部

セントロニクス→RS-232C変換部は，図1，図2のUARTおよびUARTの右側の部分です．

次に，回路の動作について説明します．

① まず，セントロニクス入力の$DATA_1$～$DATA_8$をバッファLS541を通して，UARTの送信部のTBR_1～TBR_8に加えます．

② 次に，セントロニクス入力の$\overline{\mathrm{STROBE}}$をパルス幅約0.5μsの単安定マルチバイブレータで整形して，

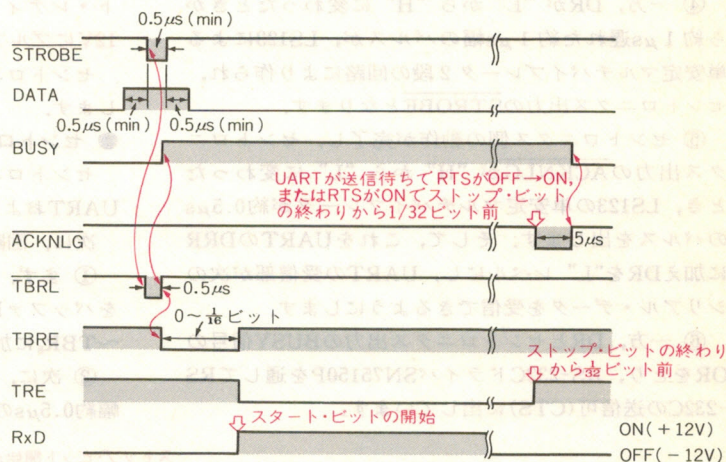
〈図3〉[11]
RS-232C→セントロニクス変換部のタイムチャート

〈表1〉
RS-232Cコネクタの信号配置

ピンNo.	信号名	方向	機能
1	FG		保安アース
2	TxD	→本器	送信データ
3	RxD	←本器	受信データ
4	RTS	→本器	送信要求(本器ではセントロニクス→RS-232C変換部のハンドシェイクに使用)
5	CTS	←本器	送信可(本器ではRS-232C→セントロニクス変換部のハンドシェイクに使用)
6	DSR	←本器	データ・セット・レディ(本器では＋12Vにプルアップ)
7	SG		信号アース
8	CD	←本器	キャリア検出(本器では＋12Vにプルアップ)

(注)本器の信号配置はモデム定義になっている

〈表2〉
セントロニクス出力コネクタの信号配置

信号ピンNo.	リターン側ピンNo.	信号名	方向	解説
1	19	$\overline{\text{STROBE}}$	出	定常"H"．パルス幅約1μs
2	20	$DATA_1$	出	8ビットのデータ信号線．ストローブ信号の"L"パルスの前約1μsから$\overline{\text{ACKNLG}}$が"⤓"になるまでデータが確定している
3	21	$DATA_2$	出	
4	22	$DATA_3$	出	
5	23	$DATA_4$	出	
6	24	$DATA_5$	出	
7	25	$DATA_6$	出	
8	26	$DATA_7$	出	
9	27	$DATA_8$	出	
10	28	$\overline{\text{ACKNLG}}$	入	周辺装置からのデータ入力の完了を示す信号
11	29	BUSY	入	周辺装置が入力の可否を示す
16		0 V		ロジックGNDレベル
17		CHASSIS-GND		筐体グラウンド
33		GND		

〈図4〉
セントロニクス→RS-232C変換部のタイムチャート

UARTの送信部の$\overline{\text{TBRL}}$端子に加えます．UARTでは$\overline{\text{TBRL}}$入力が"H"→"L"→"H"と変化する間のうち"L"から"H"に変化するときにパラレルからシリアルへの変換が開始されるのに対して，標準セントロニクスでは，$\overline{\text{STROBE}}$が"H"から"L"に変化するときにデータの読み込みが行われると規定されています．

そのため，タイミングのずれによるデータの読み込みミスが生じるのを防ぐために，単安定マルチバイブレータを用いています．

③ この$\overline{\text{TBRL}}$の立ち上がりにより，セントロニクスのパラレル・データがシリアル・データとなり，UARTのTRO端子に出力され，RS-232CドライバSN75150Pを通してRS-232Cの受信データ(RxD)に出力されます．

④ UARTのビジィ状態は，UARTのTREとTBREのNANDで表されます．RS-232Cの送信要求(RTS)がONの場合は，このビジィ状態を表す信号の立ち下がりでパルス幅約5μsの単安定マルチバイブレータを働かせ，セントロニクス入力の$\overline{\text{ACKNLG}}$とします．また，UARTが送信待ちの場合は，RTSがOFFからONに変わったときにこの単安定マルチバイブレータを働かせ，セントロニクス入力の$\overline{\text{ACKNLG}}$とします．

⑤ セントロニクス入力のBUSY信号は，$\overline{\text{TBRL}}$の立ち上がりから$\overline{\text{ACKNLG}}$の立ち上がりまでの間"H"になるようになっています．

このセントロニクス→RS-232C変換部のタイムチャートを図4に示します．

本変換部では，動作に必要なRS-232Cの信号は，受信データ(RxD)，送信要求(RTS)および信号接地(SG)のみです．

RTSによる制御法は，ソード社のM223のモデム定義のRS-232Cコネクタを参考にしました．

セントロニクス入力コネクタの信号配置を表3に示します．

◆ 使用例

本器はセントロニクス，RS-232C両入力対応のPTPインターフェースの一部分として製作されたも

〈表3〉
セントロニクス入力コネクタの信号配置

信号ピンNo.	リターン側ピンNo.	信号名	方向	解説
1	19	$\overline{STROBE}$	入	定常"H"."┐_"から約0.5μs後にデータ読み込み．パルス幅0.5μs以上
2	20	$DATA_1$	入	8ビットのデータ信号線．ストローブ信号の"L"パルスの前後0.5μs間はデータが確定していなければならない
3	21	$DATA_2$	入	
4	22	$DATA_3$	入	
5	23	$DATA_4$	入	
6	24	$DATA_5$	入	
7	25	$DATA_6$	入	
8	26	$DATA_7$	入	
9	27	$DATA_8$	入	
10	28	$\overline{ACKNLG}$	出	データ入力の完了を示す，割り込み用5μs
11	29	BUSY	出	本器が次のデータの入力の可否を示す，"L"で可
12		PE	出	"H"で用紙なし．"L"に設定してある
16		0V		ロジックGNDレベル
17		CHASSIS-GND		筐体グラウンド
32		$\overline{ERROR}$	出	"H"に設定してある
33		GND		

ので，RS-232C→セントロニクス変換部は，ソードM223とセントロニクス入力PTPあるいは，エプソン社のプリンタTP80〔またはFP100(現在のVPシリーズの前がMPシリーズで，その前のシリーズにあたる)〕との間に入れて，ボーレート9600で良好に動作中です．

また，エプソン社のハンドヘルド・パソコンのHC20とプリンタFP80との間に入れて，ボーレート4800で良好に動作しました．

セントロニクス→RS-232C変換部は，データ・ジェネラル社のミニコンNOVA 3のライン・プリンタ出力(セントロニクス入力のプリンタと接続できるように信号の論理を変更してある)とソード社のM23との間に入れて，NOVA 3上で開発したプログラムやデータ・ファイルをM23に移植するのに用いました．

移植に際しては，ソフトウェア開発の負担はまったくといってよいほどなく，NOVA 3側では，プログラムやデータをライン・プリンタに印字する命令を与えるのみです．M23側では，エディタでRS-232C入力(ボーレート1200に設定，RTS信号による制御は有効でない)を入力ファイルに指定し，データを入力し，フロッピ・ディスク上の出力ファイルに書くだけです．

また，ソードM23のセントロニクス・プリンタ出力とエプソンHC20との間に入れて，BASICのプログラムを転送してみました．ボーレート300まで良好に動作しましたが，ボーレート600以上でHC20側のバッファ・オーバフロー・エラーが起こり，動作しませんでした．これは，RS-232Cの制御信号RTSによるハンドシェイクが行われないために，変換器からのRS-232C出力がたれ流し状態になっているためです．

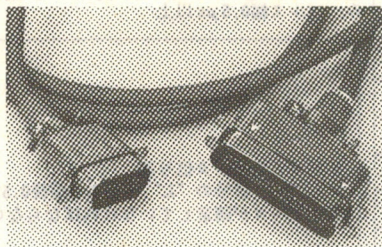
◆ 使用上の注意点

本器のようにセントロニクス↔RS-232C変換する場合，両インターフェースの動作に起因する問題点があります．それは，セントロニクスがハンドシェイクで動作するのに対応して，RS-232Cもハンドシェイクで動作する必要があるのに，現実にはRS-232Cは必ずしも制御信号線を用いたハンドシェイクを行って使われているとは限らないことにあります．

このため，上で述べた使用例以外では，原理的に本器が使用できない場合があることも考えられます．また，原理的には可能でも信号線の接続法の違いにより動作しない場合もありますので，個々のパソコンおよび周辺装置のタイミング，信号線の役割などについて十分検討の上使用してください．

とくに，RS-232Cの場合，使用する信号線については機器によってまちまちで，その信号線の役割も様々に使われています．十分注意してください．

なお，RS-232Cのハンドシェイクの問題は，ソフトウェアでRTSやDTRなどの制御信号をコントロールすることにより解決できる場合もあります．

§1-4

簡易型プリンタ・バッファの製作

斉藤洋司/生沼守英

最近の半導体技術の進歩は著しく，とくにICメモリの高密度化が進んでいます．そのおかげで，パソコンの記憶容量は飛躍的に増大しました．しかし，プリンタなどの出力装置の高速化はあまり望めず，ユーザの要求に十分対応しきれていないのが現状です．

我々の研究室においても，測定データをプリンタやプロッタへ出力する際，それらの動作速度の遅さが気になっていました．この問題を解決し得るのが，並列処理により高速化を行うプリンタ・バッファ[13]です．ただし，我々の用途ではそれほど大容量のバッファは必要でなかったことや，市販品はプリンタ切り替え機を兼ねていたりして，適当な価格の製品が見当たらなかったため，製作を試みることにしました．

比較的製作経験の浅い人でも製作できるように，再現性を重視しながら可能な限り回路を簡素化しました．ただし，少なくともテスタとオシロスコープは，デバッグのために必要になるでしょう．

なお，本機はプリンタ切り替え機を兼ねたプリンタ・バッファであり，セントロニクス[14]準拠のインターフェースを備えたパソコンやプリンタ，プロッタなどに使用できます．

本機の回路構成

図1に本機の全回路図を示します．Z80 CPUを中心としてRAM, ROM, PPIなどにより構成しています．次に，それらの部品の選択および使い方について簡単に説明します．

● メモリ

まずメモリですが，RAMには再現性を高めるためスタティックRAM(SRAM)を用いることにしました．その場合，必要に応じて56Kバイトまでの範囲で，8Kバイトごとにバッファ容量を増減することができます．

回路構成を変えずに，さらにコストを抑えたい場合は，疑似スタティックRAM(PSRAM)[15]を用いるとよいでしょう．PSRAMは，ピン配置がSRAMと同じで，使い方も同様にできますが，内部はダイナミックRAMですので，リフレッシュのため若干の回路変更を要します．

256KビットPSRAM(HM65256：日立)の場合，パッケージのピン数の制限から，64KビットPSRAMにはある$\overline{RFSH}$端子のような端子がありません．そのため，図2に示すタイミングで$\overline{OE}$および$\overline{CS}$端子にパルスを加えることにより，自動的にリフレッシュが行われるようになっています(これをオート・リフレッシュ・モードと呼ぶ)．

本機では，図1の赤枠内のRAM_1～RAM_4の代わりに，図3の回路を用いればよいのですが，LS00とLS08を追加する必要があります．

ROMには，2764または27128を用いました．プログラム・サイズは500バイト以下ですので，ほんの一部しか使用していないことになります．

● PPI 8255A

次にPPIについてですが，8255Aをデータ入出力のポートとして用いています．各ポートの割り当ては，表1を参照してください．

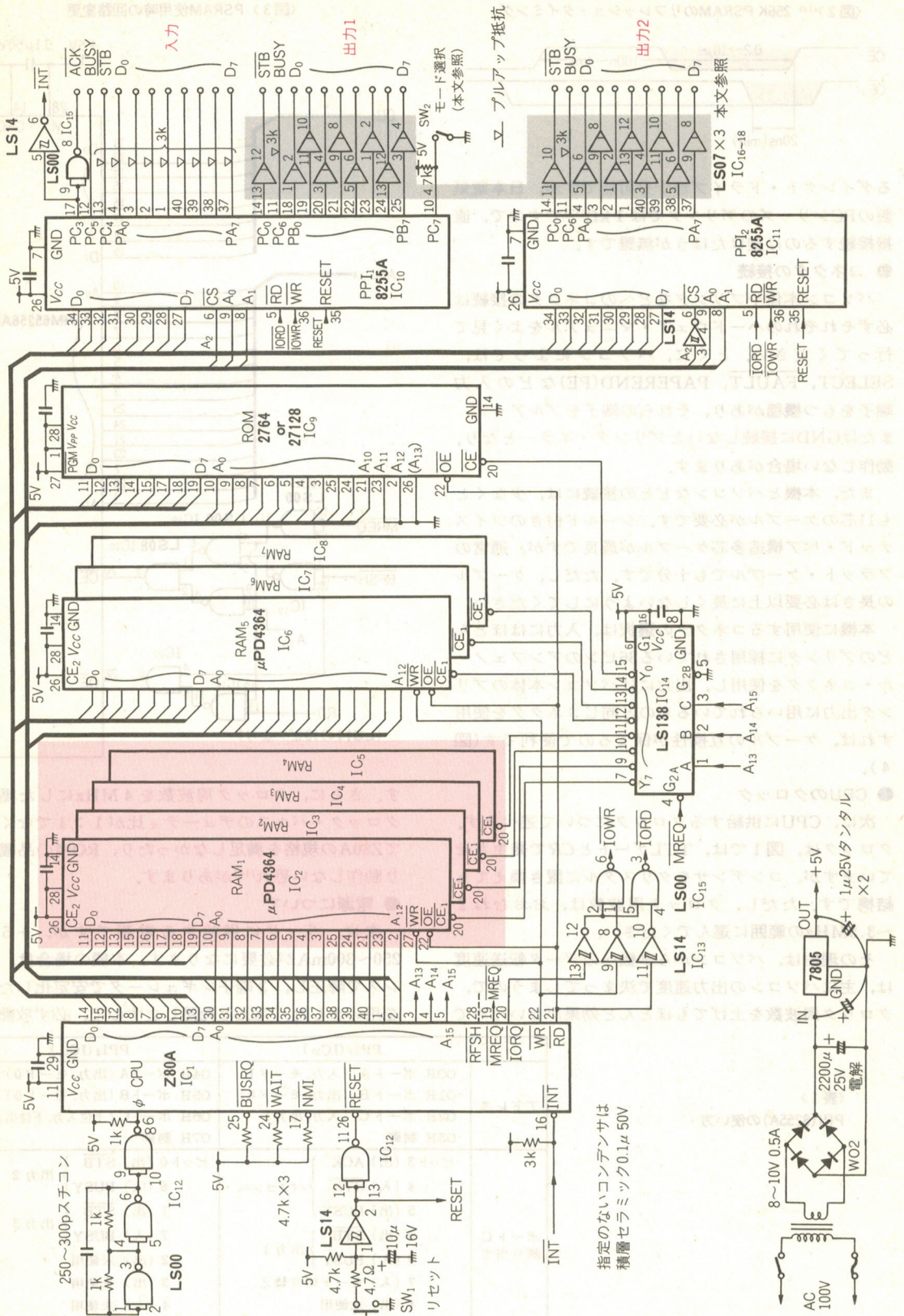
データ入力ポートのみ，モード1[16]で使用され，パソコンから$\overline{STB}$信号が入ると，ハード的に$\overline{BUSY}$および割り込み要求信号が発生するようになっています．出力ポートはすべてモード0で，$\overline{STB}$の制御はビットのセット/リセットにより行います．なお，$\overline{ACK}$は無視しています．

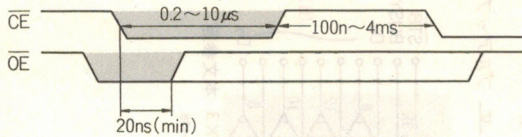
セントロニクス仕様では，LS14相当のシュミット・トリガを抵抗でプルアップした回路を入力とし，7406などのオープン・コレクタで出力するのが原則となっています．

しかし，とくにノイズ環境が悪くない限り，図1のように8255Aで直接入力を行っても問題ありません．

出力ドライバの7407も不要な場合が多いのですが，8255Aのシンク能力が2.5mAと小さいため，プリンタなどの入力が2kΩ以下でプルアップされている場合は，このドライバを省略すると動作の保証はできません．

例えば，エプソン社のTP, MPシリーズのプリンタでは，プルアップ抵抗は3kΩですので，8255Aによ

〈図1〉プリント・バッファ/切り替え器の回路構成

〈図2〉[14] 256K PSRAMのリフレッシュ・タイミング

〈図3〉PSRAM使用時の回路変更

るダイレクト・ドライブが十分可能ですが，日本電気製のPCシリーズのプリンタでは1kΩ[17]ですので，直接接続するのは避けたほうが無難です．

● コネクタの接続

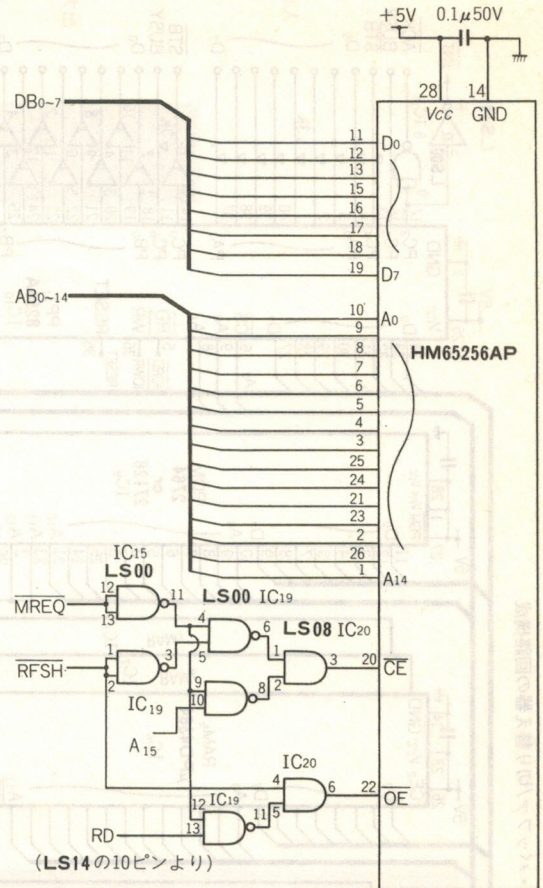
パソコン本体やプリンタなどへのコネクタの接続は，必ずそれぞれのハードウェア・マニュアルをよく見て行ってください．とくに，パソコンによっては，SELECT，$\overline{\text{FAULT}}$，PAPEREND(PE)などの入力端子をもつ機種があり，それらの端子をプルアップ，またはGNDに接続しないとプリンタ・エラーとなり，動作しない場合があります．

また，本機とパソコンなどとの接続には，少なくとも11芯のケーブルが必要です．シールド付きのツイステッド・ペア構造多芯ケーブルが最良ですが，通常のフラット・ケーブルでも十分です．ただし，ケーブルの長さは必要以上に長くしないようにしてください．

本機に使用するコネクタの選択は，入力にはほとんどのプリンタに採用されている36ピンのアンフェノール・コネクタを使用し，出力にはパソコン本体のプリンタ出力に用いられているものと同じコネクタを使用すれば，ケーブルの互換性が保てるので便利です(図4)．

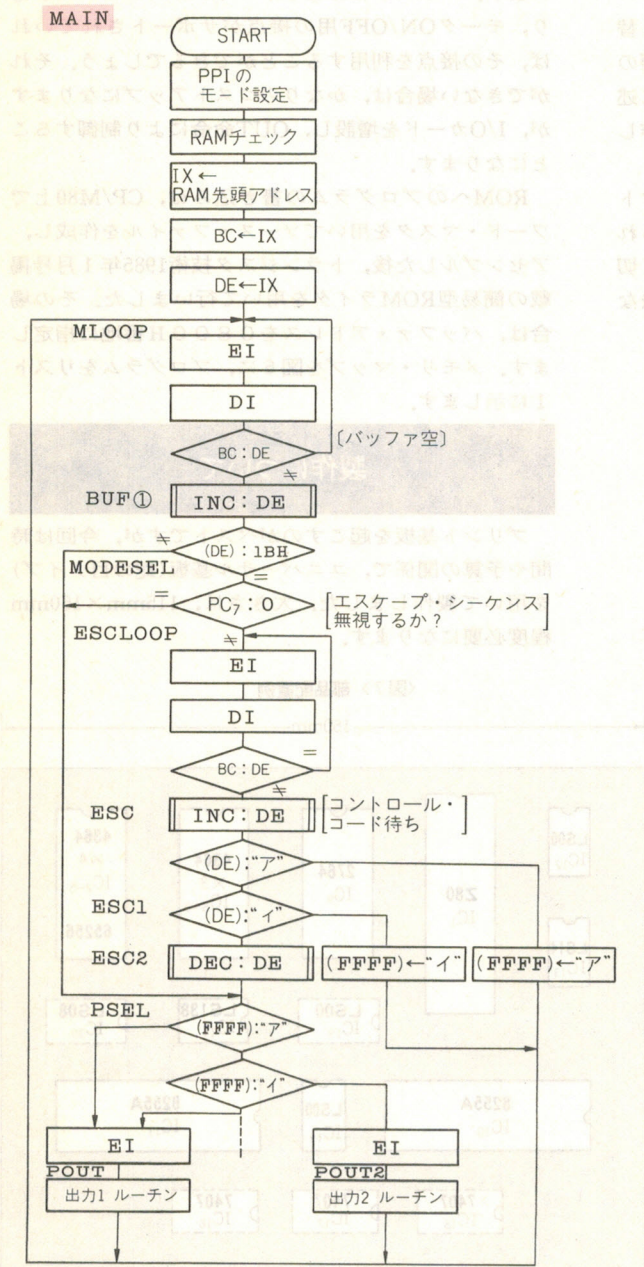
● CPUのクロック

次に，CPUに供給するクロックについて述べます．クロックは，図1では，TTLゲートと*CR*で発生させていますが，コンデンサをクリスタルに置き換えても結構です．ただし，クロック周波数は，おおむね2～3.5MHzの範囲に選んでください．

その理由は，パソコンから本機へのデータ転送速度は，主にパソコンの出力速度で決まってしまうので，クロック周波数を上げてもほとんど効果がないためです．さらに，クロック周波数を4MHzにした場合，クロック・パルスのデューティ比が1：1でなくなってZ80Aの規格を満足しなかったり，ROMの品種により動作しない恐れ[18]があります．

● 電源について

次に，ボードに供給する電源ですが，＋5V，250～300mAが必要になります．本機の場合は，トランスで降圧し，3端子レギュレータで安定化した電源を用いました．7805は発熱しますので，必ず放熱板を

〈表1〉
PPI(8255A)の使い方

	PPI1 (IC10)	PPI2 (IC11)
アドレス	00H ポートA (入力，モード1) 01H ポートB (出力，モード0) 02H ポートC (入力，出力) 03H 制御	04H ポートA (出力，モード0) 05H ポートB (出力，モード0) 06H ポートC (上位入力，下位出力) 07H 制御
ポートC割り当て	ビット3 (出) ACK } パソコンへ 4 (入) $\overline{\text{STB}}$ } 5 (出) BUSY } 0 (出) $\overline{\text{STB}}$ } 出力1 6 (入) BUSY } 7 (入) モード切り替え 1 (出) 未使用 2 (出) 未使用	ビット0 (出) $\overline{\text{STB}}$ } 出力2 6 (入) BUSY } 1 (出) $\overline{\text{STB}}$ } 出力3 7 (入) BUSY } 2 (出) 未使用 3 (出) 未使用 4 (入) 未使用 5 (入) 未使用

つけてください。

ソフトウェアについて

プログラムの流れ図を図5に示します。基本的な働きは，入力ポートに$\overline{\text{STB}}$パルスが入って，ハード的に割り込み要求が起こると，割り込み禁止状態となり，かつバッファ・メモリがあふれていなければ，データを1文字入力し，メモリに格納することと，バッファ内のデータが空でなければデータを出力すること，および出力ポートの切り替え命令を解読することです。

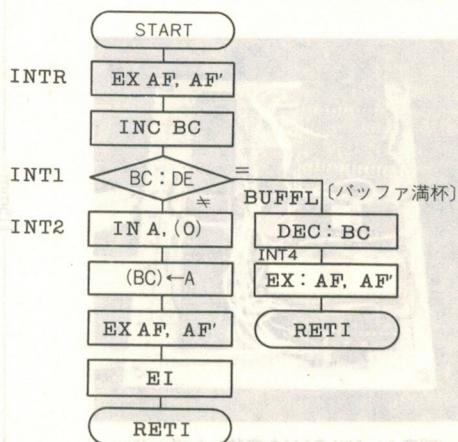
割り込みには，Z80モード1を用いました。割り込みルーチンに入ると，条件を満たせばデータをPPI_1(IC_{10})のポートAより入力し，レジスタBCの表す番地のメモリに格納します。

出力は，条件を満たせばFFFFH番地の内容が示す出力ポートに，レジスタDEの表すメモリの内容を出力するようになっています。

〈図5〉プログラムのフローチャート

〈図4〉[17] セントロニクス・インターフェースのコネクタ・ピン配置とタイミング

(a) コネクタ接続(日本電気製のプリンタの場合)

(b) ハンドシェイク・タイミング

● 出力ポートの切り替え

出力ポートの切り替え(FFFFH番地にデータを書き込む動作)は，エスケープ・シーケンスにより行います．

具体的には，1BHに続くデータがB1Hならば出力1が選択され，B2Hならば出力2が選択されます．これらのキャラクタ・コードは，カタカナの“ア”または“イ”に相当し，プリンタでは機能コードとして使用されることはまずありません．また，このエスケープ・シーケンスは，現在の国産のパソコンでは必ずサポートしていると考えられます．

ただし，このエスケープ・シーケンスによる切り替え方法の欠点は，プリンタにドット・イメージ(画面のハード・コピーを含む)を出力するときに，偶然先に述べたようなデータ・パターンが現れた場合，誤動作してしまうという問題があることです．

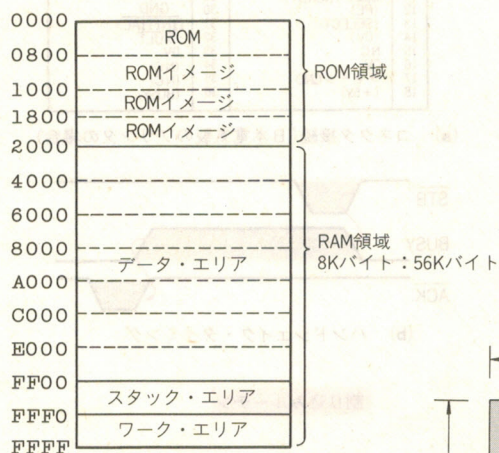
この問題をすべての機種のプリンタに対し，ソフトで解決することはほぼ不可能です．そこで，考えられる解決法としては，パソコンのI/Oポートを用いて切り替えを行う方法[19]，スイッチで切り替える方法などがあります．

〈図6〉プログラムのメモリ・マップ

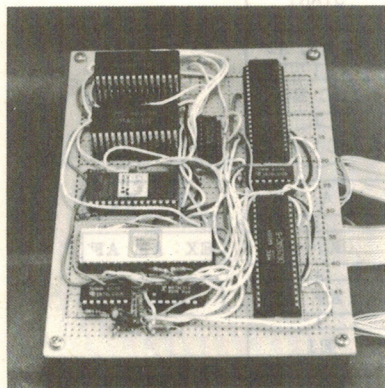
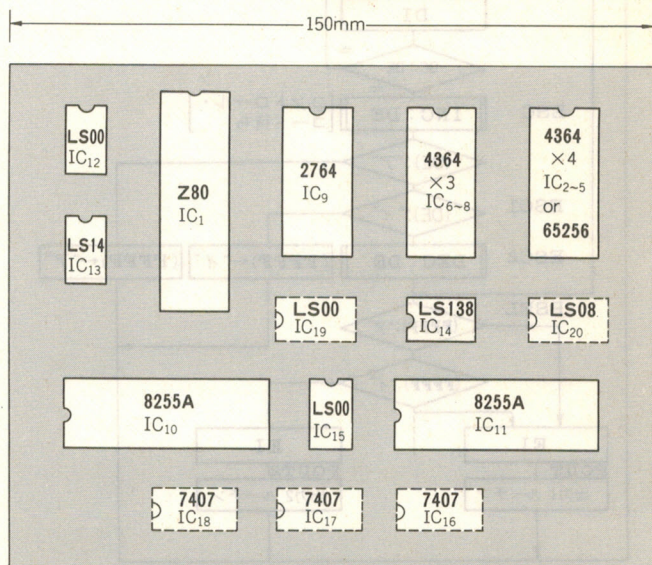
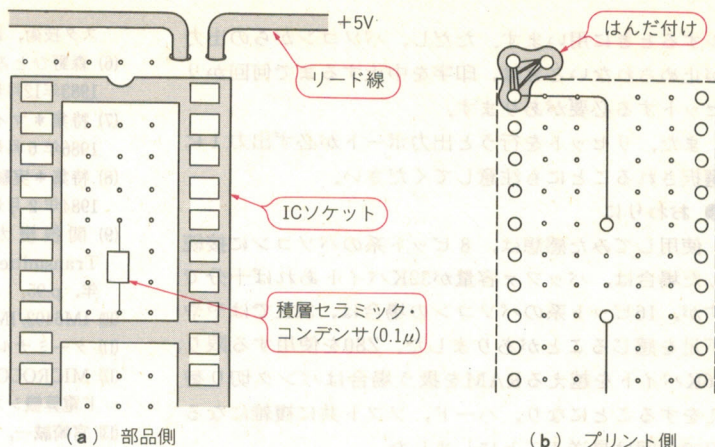
〈写真1〉56K RAMを実装したボード

本機では，その両方の方法が使えるような回路とし，今回はとりあえずスイッチで，通常モードと切り替えコードを無視するモードとに切り替えられるようにしました．実際には，ドット・イメージを出力するときは，IC_{10}の10ピンがGNDに接続されるようにSW_2を閉じてください．

パソコンのI/Oポートで，モード切り替えを行う方法は，その端子にSW_2の代わりにリレー接点を接続するか，0Vまたは5Vの信号を加えることにより行います．パソコンにカセット・インターフェースがあり，モータON/OFF用の接点がサポートされていれば，その接点を利用することができるでしょう．それができない場合は，かなりのコストアップになりますが，I/Oカードを増設し，OUT命令により制御することになります．

ROMへのプログラムの書き込みは，CP/M80上でワード・マスタを用いてソース・ファイルを作成し，アセンブルした後，トランジスタ技術1985年1月号掲載の簡易型ROMライタを用いて行いました．その場合は，バッファ・アドレスを0800H番地に指定します．メモリ・マップを図6に，プログラムをリスト1に示します．

製作について

プリント基板を起こすのがベストですが，今回は時間や予算の関係で，ユニバーサル基板(蛇の目タイプ)を用いて製作しました．大きさは，115mm×150mm程度必要になります．

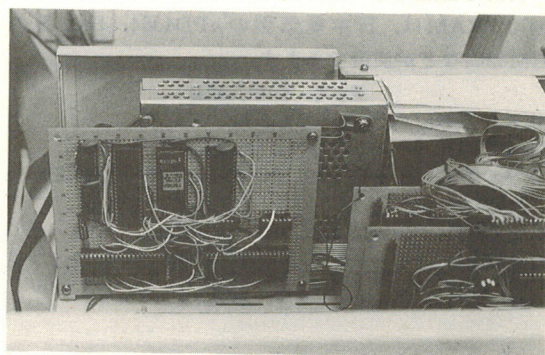
〈図7〉部品配置例

〈図8〉パスコンの実装例(RAM, ROMの場合)

部品の配置は，図7を参考にしてください．あまりお勧めできませんが，64K SRAMは，省スペース省力化のため，3～4個を上下に重ね，$\overline{CE_1}$(20ピン)以外の端子を直接はんだ付けして，24～32KバイトのRAMモジュールのようにして実装しました．

C-MOS SRAMは，発熱が極めて少ないので，信頼性には問題ないようです．なお，RAMに直接はんだ付けをするときは，リーク電流の少ない(セラミック・ヒータ使用の)はんだゴテを用い，素早く行ってください．

配線は，普通のビニル皮覆の撚り線や直径0.3mmのポリウレタン線(ポリウレタンをコーティングした銅線)をはんだ付けして行いました．＋5VとGNDライン以外はラッピング配線でもかまいませんが，ICソケットに相当のコストがかかってしまうでしょう．

配線の順序は，ICソケットを固定した後，パスコンの取り付け，＋5V，GNDライン，続いて$\overline{RD}$，$\overline{WR}$などの制御ラインの順に行います．パスコンに積層セラミック・コンデンサを用いれば，図8のようにICソケットの中に収めることができるはずです．また，＋5V, GNDラインは，太めの撚り線で短く配線してください．

最後に，アドレス，データ・バスの配線を，基板の裏面でポリウレタン線を用いて行います．ポリウレタン線は，溶けたはんだの熱で皮覆が取れるようになっていますが，その皮覆が溶けるのにやや時間を要するので注意が必要です．そこで，接続する前にポリウレタン線のはんだ付け箇所を，あらかじめはんだメッキして皮覆を溶かしておいたほうが確実です．

配線が一応終わったら，必ずテスタなどでソケットの上から導通をチェックしてください．OKならばICを正しく取り付け，＋5Vの電圧を加えてください．オシロスコープで，Z80のϕに数MHzのクロックが入力されていること，$\overline{INT}$端子が“H”になっていることを確認します．

〈写真2〉PC8012(拡張I/Oユニット)内に組み込んだ例

データ入力がないのに$\overline{INT}$が“L”になったままになっている場合は，PPI周辺に配線不良があり，PPIのモード設定がなされていないことが多いようです．

◆ 本機の使用方法

コネクタを入力，出力共に正しく接続し，電源を入れてください．このとき，パソコンやプリンタなどの電源も同時に投入しないと，機種によっては動作しなかったり，誤動作することがあります．

プリンタの切り替えは，例えばBASICでは，

LPRINT␣chr$(27)+“ア”；

または，

LPRINT␣chr$(27)+“イ”；

を実行することにより行われます．“ア”が出力1，“イ”が出力2に対応しており，電源投入時には出力1が選択されています．

ソフトの説明でも述べましたが，ビット・イメージを出力するときは，上記のコントロール・コードと誤解釈される恐れがありますので，あらかじめSW_2を閉じておいてください．その場合，出力ポートの選択はSW_2を閉じる前の状態に固定されてしまうことに注意してください．

リセットSWは，印字を中止し，データをキャンセ

ルするときに用います．ただし，パソコンからの出力が止められないときは，印字を中止するまで何回かリセットする必要があります．

また，リセットを行うと出力ポートが必ず出力１に選択されることにも注意してください．

● おわりに

使用してみた感想は，８ビット系のパソコンに接続した場合は，バッファ容量が32Kバイトあれば十分ですが，16ビット系のパソコンの場合は，32Kではやや不足を感じることがありました．Z80を使用する限り，56Kバイトを越えるRAMを扱う場合はバンク切り替えをすることになり，ハード，ソフト共に複雑になるので今回は見送ることにしました．

特殊な部品はないので簡単に入手できると思います．64K SRAMは，日本電気製のμPD4364，日立製のHM6264などが使用できます．

◆参考・引用*文献◆

(1)*スター精密，AR-2400取扱説明書．
(2)*エプソン，VP-130K，IP-130K取扱説明書．
(3)*日本電気，PC-PR201H2，PC-PR601，PC-PR406LPユーザーズマニュアル．
(4)*日本電気，PC-9801 ユーザーズマニュアル．
(5) 神崎康宏；特集＊マイコン設計技術の完全マスタ，トランジスタ技術，1985年５月号．
(6) 森野ひとみ；マイコンとデータ伝送，トランジスタ技術，1983年12月号．
(7) 特集＊マイコン周辺LSI完璧マスタ，トランジスタ技術，1986年６月号．
(8) 特集＊実験で学ぶディジタルIC回路，トランジスタ技術，1984年２月号．
(9) 関根慶太郎；Universal Asynchronous Receiver/Transmitterその構造と応用，HAM Journal，No.４，1975年，p.95．
(10) IM6402/IM6403データ・シート，インターニックス．
(11) ターミナル・プリンタEPSON TP-80仕様書，信州精器．
(12) MICROCOMPUTER M200 SERIES，３版，1980年，ソード電算機システム．
(13) 宮崎誠一，他；基本的マイコン・システムと割り込みの基礎，トランジスタ技術，1985年10月号，p. 446．
(14) 神崎康宏；マイコン設計技術の完全マスタ第５章，トランジスタ技術，1985年５月号，p. 421．
(15)*日立新製品速報，HM65256データ・シート．
(16) 日本電気，NEC半導体マニュアル「マルチチップ」．
(17)*日本電気，PC8821/PC8822ユーザーズ・マニュアル．
(18) 松本吉彦；保存版・Z80の徹底研究，トランジスタ技術，1980年11月号，p. 275．
(19) 柴田健司；プリンタ，プロッタ切り替え器の製作，トランジスタ技術，1985年４月号，p. 491．

〈リスト１〉コントロール・プログラム

```
 2: MAIN ROUTINE
 3:
 4:                         SUBTTL  MAIN ROUTINE
 5:                         TITLE   PRINTER BUFFER
 6:                         ;       PRINTER BUFFER with SELECTER
 7:                         ;               VER 3.2      1985.11.19
 8:                         ;               PRESENTED BY Y.SAITO
 9:                         ;
10:                                 .z80
11:   FEFF                  MAX     EQU     0FEFFH
12:   00FE                  HMAX    EQU     HIGH MAX
13:   00FF                  LMAX    EQU     LOW  MAX
14:   0800                  OFFSET  EQU     800H;
15:   0000'                 ASEG    ;ABSOLUTE CODE
16:                         ;
17:                                 ORG     OFFSET
18:   0800    C3 0900               JP      INIT
19:                         ;
20:                                 ORG     OFFSET+38H
21:   0838    C3 0A2A               JP      INTR
22:                         ;
23:                                 ORG     OFFSET+100H
24:   0900                  INIT:
25:   0900    3E 88                 LD      A,088H   ;PPI2MODE
26:   0902    D3 07                 OUT     (7H),A
27:   0904    3E 0F                 LD      A,0FH    ;STB2 HIGH
28:   0906    D3 06                 OUT     (6H),A
29:                         ;
30:   0908    3E B8                 LD      A,0B8H   ;PPI1 MODE
31:   090A    D3 03                 OUT     (3H),A
32:   090C    3E 09                 LD      A,9H     ;PPI EI
33:   090E    D3 03                 OUT     (3H),A
34:   0910    3E 01                 LD      A,1H     ;STB1 HIGH
35:   0912    D3 02                 OUT     (2H),A
36:                         ;
37:   0914    3E B1                 LD      A,0B1H
38:   0916    32 FFFF               LD      (0FFFFH),A;SELECT POUT1
39:   0919    31 FFF0               LD      SP,0FFF0H;SP SET
40:                         ;
41:   091C    21 FF00       RAMH:   LD      HL,0FF00H;MAX RAM ADDRES
42:   091F    36 00         R1:     LD      (HL),00H;CHECK RAM
43:   0921    7E                    LD      A,(HL)
44:   0922    FE 00                 CP      00H
45:   0924    C2 0933               JP      NZ,R2
46:   0927    36 FF                 LD      (HL),0FFH
47:   0929    7E                    LD      A,(HL)
48:   092A    FE FF                 CP      0FFH
49:   092C    C2 0933               JP      NZ,R2
52:
53:   092F    25                    DEC     H
54:   0930    C3 091F               JP      R1
55:                         ;
56:   0933    24            R2:     INC     H        ;HEAD ADRS OF RAM
57:   0934    E5                    PUSH    HL
58:   0935    DD E1                 POP     IX
59:   0937    DD E5                 PUSH    IX       ;INPUT ADRS
```

〈リスト1〉コントロール・プログラム(つづき)

```
 60:  0939    C1                    POP   BC
 61:  093A    DD E5                 PUSH  IX          ;OUTPUT ADRS
 62:  093C    D1                    POP   DE
 63:  093D    ED 56                 IM    1
 64:  093F    FB                    EI
 65:                         ;
 66:                         ;
 67:                         SUBTTL MAIN ROUTINE
 70:
 71:
 72:
 73:                         ;
 74:                         ;      MAIN ROUTINE
 75:                         ;
 76:  0940    FB             MLOOP: EI          ;WAITING DATA IN
 77:  0941    00                    NOP
 78:  0942    00                    NOP
 79:  0943    F3                    DI
 80:                         ;
 81:  0944    78                    LD    A,B
 82:  0945    BA                    CP    D
 83:  0946    C2 0951               JP    NZ,BUFO
 84:  0949    79                    LD    A,C
 85:  094A    BB                    CP    E
 86:  094B    C2 0951               JP    NZ,BUFO
 87:  094E    C3 0940               JP    MLOOP
 88:                         ;
 89:  0951    CD 099C        BUFO:  CALL  INCDE
 90:                         ;
 91:  0954    1A                    LD    A,(DE)   ;CHR=ESC?
 92:  0955    FE 1B                 CP    1BH
 93:  0957    C2 09BB               JP    NZ,PSEL
 94:  095A    C3 095D               JP    MODESEL
 95:                         ;
 96:  095D    DB 03          MODESEL:IN   A,(3H)
 97:  095F    E6 80                 AND   80H
 98:  0961    CA 09BB               JP    Z,PSEL
 99:  0964    C3 0967               JP    ESCLOOP
100:                         ;
101:  0967    FB             ESCLOOP:EI         ;WAITING CONTROL CHR
102:  0968    00                    NOP
103:  0969    F3                    DI
104:  096A    78                    LD    A,B
105:  096B    BA                    CP    D
106:  096C    C2 0977               JP    NZ,ESC
107:  096F    79                    LD    A,C
108:  0970    BB                    CP    E
109:  0971    C2 0977               JP    NZ,ESC
110:  0974    C3 0967               JP    ESCLOOP
111:                         ;
112:  0977    CD 099C        ESC:   CALL  INCDE
113:  097A    1A                    LD    A,(DE)
114:  097B    FE B1                 CP    0B1H     ;PSEL1
115:  097D    C2 0988               JP    NZ,ESC1
116:  0980    3E B1                 LD    A,0B1H
120:  0982    32 FFFF               LD    (0FFFFH),A
121:  0985    C3 0940               JP    MLOOP
122:
123:  0988    1A             ESC1:  LD    A,(DE)
124:  0989    FE B2                 CP    0B2H     ;PSEL2
125:  098B    C2 0996               JP    NZ,ESCE
126:  098E    3E B2                 LD    A,0B2H
127:  0990    32 FFFF               LD    (0FFFFH),A
128:  0993    C3 0940               JP    MLOOP
129:                         ;
130:  0996    CD 09AD        ESCE:  CALL  DECDE
131:  0999    C3 09BB               JP    PSEL
132:                         ;
133:                         ;SUB INCDE
134:                         ;
135:  099C    13             INCDE: INC   DE
136:  099D    7A                    LD    A,D
137:  099E    FE FE                 CP    HMAX
138:  09A0    C2 09AC               JP    NZ,SUBEND
139:  09A3    7B                    LD    A,E
140:  09A4    FE FF                 CP    LMAX
141:  09A6    C2 09AC               JP    NZ,SUBEND
142:  09A9    DD E5                 PUSH  IX
143:  09AB    D1                    POP   DE
144:  09AC    C9             SUBEND: RET
145:                         ;
146:                         ;SUB DECDE
147:                         ;
148:  09AD    DD E5          DECDE: PUSH  IX
149:  09AF    E1                    POP   HL
150:  09B0    B7                    OR    A
151:  09B1    ED 52                 SBC   HL,DE
152:  09B3    C2 09B9               JP    NZ,DEC1
153:  09B6    11 FEFF               LD    DE,MAX
154:  09B9    1B             DEC1:  DEC   DE
155:  09BA    C9                    RET
156:                         ;
157:                         ;OUTPUT SELECTION
158:                         ;
159:  09BB    3A FFFF        PSEL:  LD    A,(0FFFFH)
160:  09BE    FE B1                 CP    0B1H
161:  09C0    CA 09D6               JP    Z,POUT
162:  09C3    3A FFFF               LD    A,(0FFFFH)
163:  09C6    FE B2                 CP    0B2H
164:  09C8    CA 09F2               JP    Z,POUT2
165:  09CB    3A FFFF               LD    A,(0FFFFH)
168:  09CE    FE B3                 CP    0B3H
169:  09D0    CA 0A0E               JP    Z,POUT3
170:  09D3    C3 09D6               JP    POUT
171:
172:                         ;
173:                         ;PRINT OUT CH-1
174:                         ;
175:  09D6    FB             POUT:  EI
176:  09D7    DB 02                 IN    A,(2H)   ;BUSY
177:  09D9    E6 40                 AND   40H
178:  09DB    C2 09D6               JP    NZ,POUT
179:                         ;
180:  09DE    1A                    LD    A,(DE)   ;DATA OUT
181:  09DF    D3 01                 OUT   (01H),A
182:                         ;
```

```
183:  09E1     3E 00                 LD     A,0H     ;STROBE ACTIVE
184:  09E3     D3 02                 OUT    (2H),A   ;PC OUT
185:  09E5     3E 0A                 LD     A,0AH    ;STB TIMER 1=1MI
CRO SEC.
186:  09E7     3D          TIMER:    DEC    A
187:  09E8     C2 09E7               JP     NZ,TIMER
188:  09EB     3E 01                 LD     A,1H     ;STB INACTIVE
189:  09ED     D3 02                 OUT    (2H),A
190:                       ;
191:  09EF     C3 0940               JP     MLOOP
192:                       ;
193:                       ;PRINT OUT CH-2
194:                       ;
195:  09F2     FB          POUT2:    EI
196:  09F3     DB 06                 IN     A,(6H)   ;BUSY
197:  09F5     E6 40                 AND    40H
198:  09F7     C2 09F2               JP     NZ,POUT2
199:                       ;
200:  09FA     1A                    LD     A,(DE)   ;DATA OUT
201:  09FB     D3 04                 OUT    (04H),A
202:                       ;
203:  09FD     3E 00                 LD     A,0H     ;STB ACTIVE
204:  09FF     D3 07                 OUT    (7H),A
205:  0A01     3E 0A                 LD     A,0AH    ;STB TIMER
206:  0A03     3D          TIMER2:   DEC    A
207:  0A04     C2 0A03               JP     NZ,TIMER2
208:  0A07     3E 01                 LD     A,1H     ;STB INACTIVE
209:  0A09     D3 07                 OUT    (7H),A
210:  0A0B     C3 0940               JP     MLOOP
211:                       ;
212:                       ;PRINT OUT CH-3
213:                       ;
214:  0A0E     FB          POUT3:    EI
217:
218:  0A0F     DB 06                 IN     A,(6H)   ;BUSY
219:  0A11     E6 80                 AND    80H
220:  0A13     C2 0A0E               JP     NZ,POUT3
221:                       ;
222:  0A16     1A                    LD     A,(DE)   ;DATA OUT
223:  0A17     D3 05                 OUT    (05H),A
224:                       ;
225:  0A19     3E 02                 LD     A,2H
226:  0A1B     D3 07                 OUT    (7H),A
227:  0A1D     3E 0A                 LD     A,0AH
228:  0A1F     3D          TIMER3:   DEC    A
229:  0A20     C2 0A1F               JP     NZ,TIMER3
230:  0A23     3E 03                 LD     A,3H
231:  0A25     D3 07                 OUT    (7H),A
232:  0A27     C3 0940               JP     MLOOP
233:                       ;
234:                       ;
235:                       SUBTTL    INTERRUPT ROUTINE
241:                       ;
242:                                 INTERRUPT ROUTIME
243:                       ;
244:  0A2A     08          INTR:     EX     AF,AF'
245:  0A2B     03                    INC    BC
246:  0A2C     78                    LD     A,B      ;CP BC:M-ADRS
247:  0A2D     FE FE                 CP     HMAX
248:  0A2F     C2 0A3E               JP     NZ,INT1
249:  0A32     79                    LD     A,C
250:  0A33     FE FF                 CP     LMAX
251:  0A35     C2 0A3E               JP     NZ,INT1
252:  0A38     DD E5                 PUSH   IX
253:  0A3A     C1                    POP    BC
254:  0A3B     C3 0A3E               JP     INT1
255:                       ;
256:  0A3E     78          INT1:     LD     A,B      ;CP      BC:DE
257:  0A3F     BA                    CP     D
258:  0A40     C2 0A4B               JP     NZ,INT2
259:  0A43     79                    LD     A,C
260:  0A44     BB                    CP     E
261:  0A45     C2 0A4B               JP     NZ,INT2
262:  0A48     C3 0A52               JP     BUFFL
263:                       ;
264:  0A4B     DB 00       INT2:     IN     A,(0)    ;DATA INPUT
265:  0A4D     02                    LD     (BC),A
266:  0A4E     08                    EX     AF,AF'
267:  0A4F     FB                    EI
268:  0A50     ED 4D                 RETI
269:                       ;
270:  0A52     DD E5       BUFFL:    PUSH   IX       ;BUFFER FULL
271:  0A54     E1                    POP    HL
272:  0A55     B7                    OR     A
273:  0A56     ED 42                 SBC    HL,BC
274:  0A58     C2 0A64               JP     NZ,INT4
275:  0A5B     C3 0A5E               JP     INT3
276:                       ;
277:  0A5E     01 FEFF     INT3:     LD     BC,MAX
278:  0A61     C3 0A64               JP     INT4
279:                       ;
280:  0A64     0B          INT4:     DEC    BC
281:  0A65     08                    EX     AF,AF'
282:  0A66     ED 4D                 RETI
283:                       ;
284:                                 END
288:  Macros:
289:
290:  Symbols:
291:  BUFO     0951    BUFFL    0A52    DEC1     09B9    DECDE    09AD
292:  ESC      0977    ESC1     0988    ESCE     0996    ESCLOO   0967
293:  HMAX     00FE    INCDE    099C    INIT     0900    INT1     0A3E
294:  INT2     0A4B    INT3     0A5E    INT4     0A64    INTR     0A2A
295:  LMAX     00FF    MAX      FEFF    MLOOP    0940    MODESE   095D
296:  OFFSET   0800    POUT     09D6    POUT2    09F2    POUT3    0A0E
297:  PSEL     09BB    R1       091F    R2       0933    RAMH     091C
298:  SUBEND   09AC    TIMER    09E7    TIMER2   0A03    TIMER3   0A1F
299:
300:
301:
302:  No  Fatal error(s)
303:
304:
```

§2-1 RS-232Cインターフェースの基礎

里　和政

最近多くなってきたパソコン通信は，電話回線にモデムを接続することによって行い，パソコンとモデムは，RS-232Cインターフェースで接続します．

またほとんどのパソコンには，RS-232Cインターフェースが標準装備になっていますので，簡単にシリアル伝送が行えます．

● RS-232Cとは

このインターフェースは，米国のEIAがCCITTのV.24, V.28勧告にしたがって定めたシリアル・インターフェースの規格で，信号線の電気的仕様，信号線の種類およびその機能，機械的特性(コネクタ仕様)などから成っています．そのため，伝送制御手順(通信プロトコル)は，とくに定められていません．

その電気的特性を表1に，信号線の説明を図1に示します．

規格での最大電源電圧は，±15Vですが，実際ではほとんど±12Vを使用しています．従って±10Vぐらいが信号電圧となります．

● 電圧レベル変換ICが必要

この特性を満足する，専用のドライバ/レシーバICがあります．つまり，TTLレベル(0～5V)をRS-232Cの規定しているレベルへ，電圧を変換するICです．

代表としてドライバには，TI社のSN75188，レシーバにはSN75189Aがあります．図2(a)に構成，図(b)に接続法を示します．

SN75189Aのコントロール端子は，ノイズ除去のコンデンサを付けます．オープンの状態で6Vの10nsのノイズが除去されます．コンデンサは，12pFから500pFものを使用し，6Vのノイズの場合15ns～1000ns幅で除去できます．

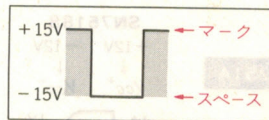
最近では，単一5Vで動作するドライバも出ています．

通信手順

伝送制御手順(プロトコル)は，通信するコンピュータ間で同一の手順を使用しなければ，正しく通信することができません．

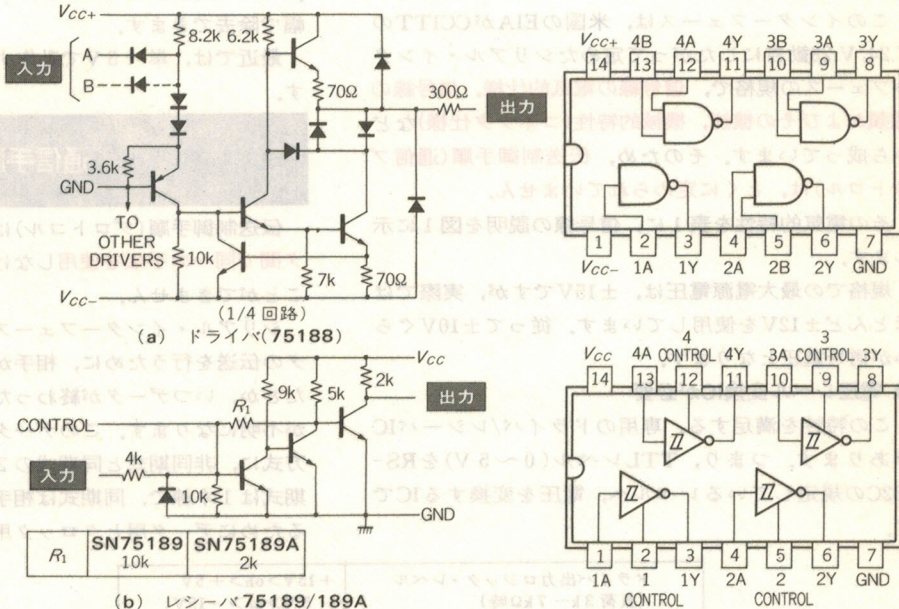
シリアル・インターフェースでは，1本の線でデータの伝送を行うために，相手がいつデータを送ってきたとか，いつデータが終わったかの伝送のタイミングが不明になります．このデータのタイミングを決める方式に，非同期式と同期式の2種類があります．非同期式は1本線で，同期式は相手の伝送タイミングを取るためにデータ用とクロック用の2本線で行います．

〈表1〉[4]
RS-232Cの電気的特性

項目	特性
ドライバ出力ロジック・レベル(負荷3k～7kΩ時)	+15V>oh>+5V −5V>ol>−15V
ドライバ出力電圧(開放時)	|Vo|<25V
ドライバ出力インピーダンス(電源断時)	Ro>300Ω
出力回路電流(短絡時)	|Io|<0.5A
ドライバ・スルーレート(立ち上がり特性)	dv/dt<30V/μs
レシーバ入力インピーダンス	7kΩ>Rin>3kΩ
レシーバ入力電圧	±15V(ドライバに同じ)
入力開放時のレシーバ出力	マーク("1")
+3V入力時のレシーバ出力	スペース("0")
−3V入力時のレシーバ出力	マーク("1")
ロジック"0"=スペース=制御ON	+15V～+5V
ノイズ・マージン(雑音余裕度)	+5V～+3V
過渡領域(入力電圧の無効域)	+3V～−3V
ノイズ・マージン	−3V～−5V
ロジック"1"=マーク=制御OFF	−5V～−15V

〈図1〉[4]
信号線の説明

名称	略号			ピン番号	8251Aに接続されるピンの名称
	RS-232C	CCITT	JIS	(25ピン・コネクタ)	
保安用接地	AA	101		1	
信号用接地	AB	102	SG	7	
送信データ	BA	103	SD	2	TxD
受信データ	BB	104	RD	3	RxD
送信要求	CA	105	RS	4	RTS
送信可	CB	106	CS	5	CTS
データ・セット・レディ	CC	107	DR	6	DSR
データ端末レディ	CD	108/2	ER	20	DTR
被呼表示	CE	125	CI	22	
データ・チャネル受信キャリア検出	CF	109	CD	8	
データ信号品質検出	CG	110	SQD	21	
データ信号速度選択	CH/CI	111	SRS	23	
送信信号エレメント・タイミング*	DA/DS	113/114	ST_1/ST_2	24/15	
受信信号エレメント・タイミング	DD	115	RT	17	
従局送信データ	SBA	118	BSD	14	
従局受信データ	SBB	119	BRD	16	
従局送信要求	SCA	120	BRS	19	
従局送信可	SCB	121	BCS	13	
従局受信キャリア検出	SCF	122	BCD	12	

(＊) 送信信号エレメント・タイミングには，データ端末装置→モデム(DA)と，モデム→データ端末(DS)の2種類がある

〈図2(a)〉[15]
代表的RS-232Cドライバ/レシーバ

〈図2(b)〉 SN75188とSN75189Aの接続図

〈図3〉非同期式伝送のフォーマット

〈図4〉代表的な非同期式の仕様例

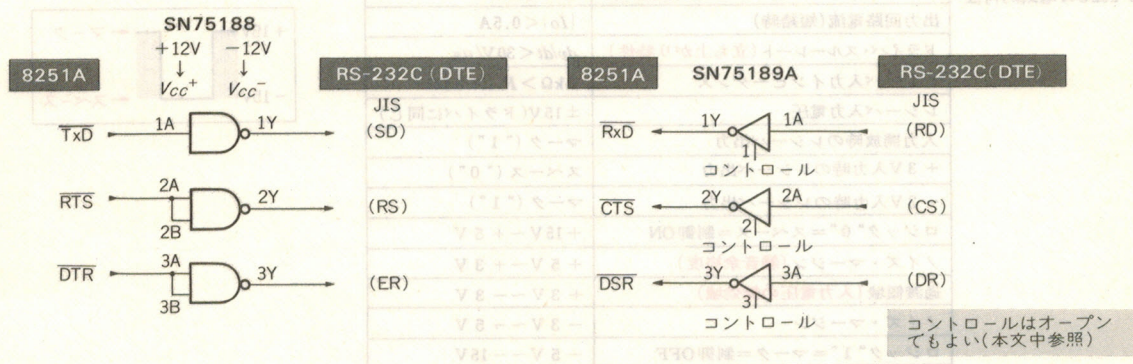
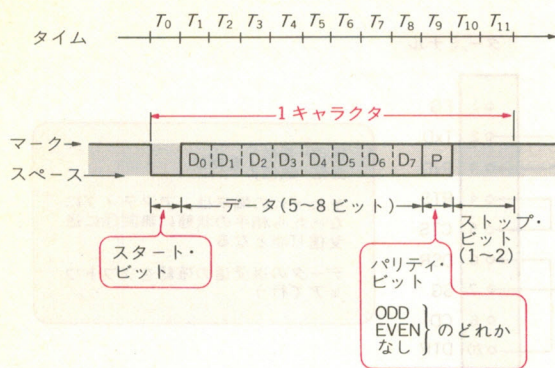
機能	仕様
ボーレート	75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200
文字長	5, 6, 7, 8ビットのいずれか
パリティ	ODD, EVEN, なしのいずれか
ストップ・ビット	1, 1.5, 2ビットのいずれか
通信方式	全二重　または　半二重

◈ 非同期式(ASYNC : Asynchronous Communication)

図3に非同期式の伝送フォーマットを示します.

シリアル伝送の場合，基本となる同期クロックが必要になりますが，このクロックをもとにして伝送するデータのタイミングを決定します.

この伝送のタイミング(速度)を，ボーレートと呼びます. ボーレートは，1秒間に何ビットのデータを送れるかの数を示します. 300ボーでは，1秒間に300ビット，1200ボーでは，1200ビットとなります.

この方式では，1キャラクタごとに同期を取るために，スタート/ストップ・ビットがキャラクタごとに付加されます.

スタート・ビットは1ビットですが，ストップ・ビットは1, 1.5, 2ビットの場合があります. 図4に非同期式の仕様を示します.

したがって，非同期式の場合には常に2〜4ビットの余分なビットがキャラクタごとに付加されるために，伝送効率が悪くなります.

しかし，この方式では，無手順と呼ぶたれ流しの方法を用いると，伝送の手順を簡単にすることができます.

また，非同期式のことを調歩同期式とも呼びます.

図5(a)〜(c)にターミナル(DTE)とターミナル(DTE)の接続を示します. ときには，この接続をT-T結線と呼びます.

図5(d)は，モデム(DCE)とターミナル(DTE)の接続を示します. この接続は，M-T結線と呼びます.

M-T結線の場合は，ストレートの接続になりますが，モデムからの信号線の出し方に特徴があり，SD信号は，ターミナルでは出力となっていますが，モデムでは入力になり，伝送コントローラのRxD(受信)に接続されます. 以下の信号も同様に入出力がターミナルの逆になっています.

とくにこの部分で接続の問題が発生しますから，接続する相手のRS-232Cの型式がDTEかDCEかを調べる必要があります.

◈ 同期式(SYNC : Synchronous Communication)

図6に同期式の伝送フォーマットを示します.

同期式では，非同期式では必要だったスタート/ストップ・ビットがなくなるため，伝送効率が上がります.

しかし，同期を取るために同期キャラクタ(SYN)がデータとは別に必要になります. 同期キャラクタは，1キャラクタごとに付加するのではなく，データ・ブロックごとに付加します. 一度同期を取ることによって，そのブロックが終了するまで同期はとれています. このブロックの長さは，256バイトまでにしている場合が多いようです. それ以上のときは，複数ブロックに分けて送信します.

● バイシンクの手順

同期式の通信プロトコルとして，バイシンク(BSC)がよく使用されますが，同期コードが2文字以上必要です.

受信側では，同期キャラクタを受け取って，同期を確立してから実データの受信を行います.

一度確立した同期は，ブロックの終わりまで保持されますが，途中でデータが途切れた場合，同期は保証されません. そのために，途中でデータが切れるときには，同期を保持するためにSYNコードを送ります. このSYNコードをタイム・フィラ(Time Filler)と呼びます. 受信側では，このSYNコードを除去します.

この方式では，伝送を制御コードによって制御する必要があります. 図7に同期式のブロック・フォーマットを，図8に制御コードの一覧を示します.

しかし，制御コードを使用しているために，すべての文字を送ることができません. そのために，このような文字も送ることのできる透過モードがあります.

透過モードは，伝送制御コードの前にDLEコードを付加して，制御コードとは区別します. DLEコードをデータとして送るときは，DLE, DLEとします.

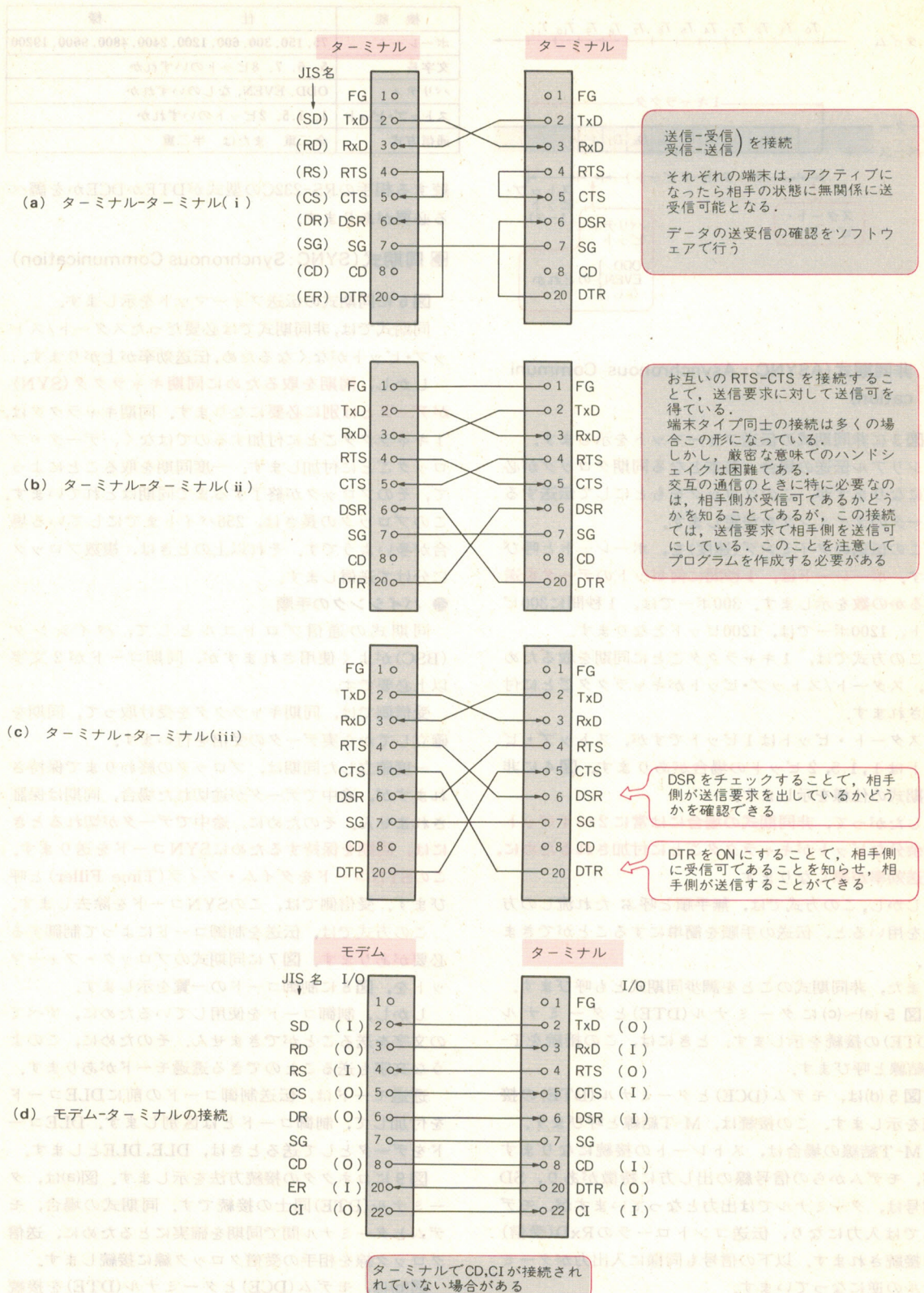
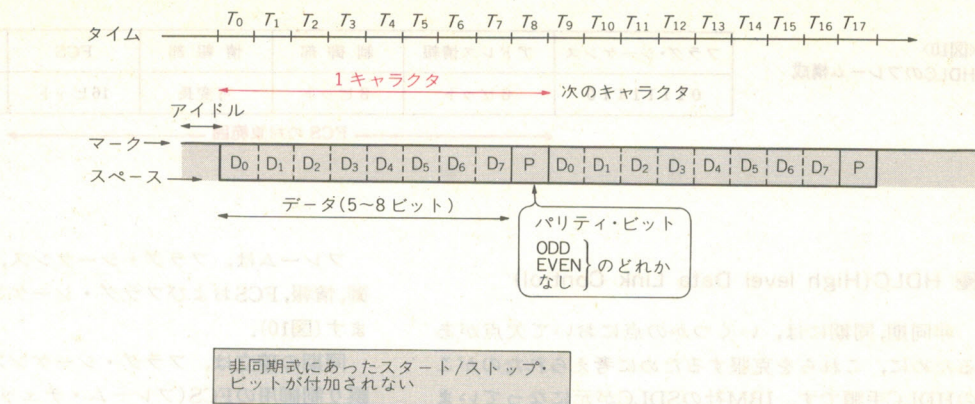
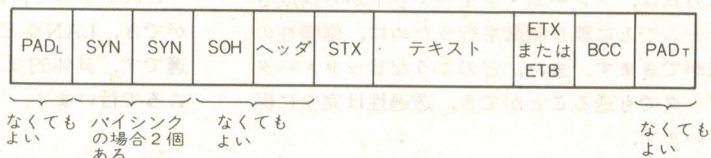
図9にコネクタの接続方法を示します. 図(a)は，ターミナル(DTE)同士の接続です. 同期式の場合，モデムとターミナル間で同期を確実にとるために，送信クロック線を相手の受信クロック線に接続します.

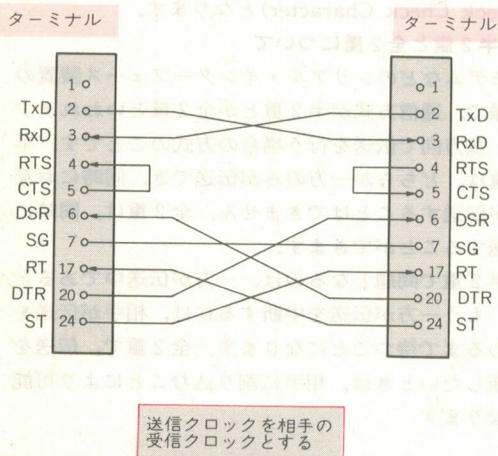
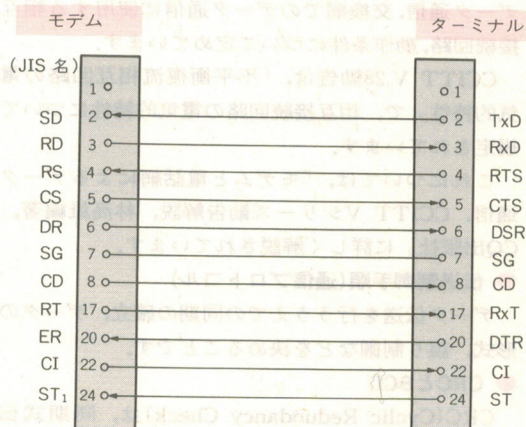
図(b)は，モデム(DCE)とターミナル(DTE)を接続する場合です.

〈図5〉ターミナルとモデム/ターミナルの接続例

〈図6〉同期式の伝送フォーマット

PAD_L	SYN	SYN	SOH	ヘッダ	STX	テキスト	ETX または ETB	BCC	PAD_T

なくてもよい（PAD_L）　バイシンクの場合2個ある（SYN）　なくてもよい（SOH・ヘッダ）　なくてもよい（PAD_T）

〈図7〉同期式のブロック・フォーマット

〈図8〉同期式制御コードの一覧

制御コード	説明	ASCII	EBCDIC	説明
SYN	同期の確立	16H	32H	同期を確立して伝送を開始する
DLE	制御コードの拡張	10H	10H	透過モードなどの拡張制御コード
SOH	ヘッダの始め	01H	01H	局番などのヘッダ部の開始を示す
STX	テキストの始め	02H	02H	データ・テキストの開始を示す
ETB	ブロックの終わり	17H	26H	ブロックの終わり(テキストの集まり)
ETX	テキストの終わり	03H	03H	データ・テキストの終わり
EOT	伝送の終わり	04H	37H	伝送が終了したことを示す
ENQ	接続要求	05H	2DH	ターミナルの接続要求
NAK	否定応答	15H	3DH	伝送に対しての否定応答
ACK_0	偶数ブロックの肯定応答	10H・30H	10H・70H	伝送に対しての肯定応答
ACK_1	奇数ブロックの肯定応答	10H・31H	10H・61H	伝送に対しての肯定応答

〈図10〉
HDLCのフレーム構成

フラグ・シーケンス	アドレス情報	制御部	情報部	FCS	フラグ・シーケンス
01111110	8ビット	8ビット	可変長	16ビット	01111110

←FCSの対象範囲→

◆ HDLC(High level Data Link Control)

非同期,同期には,いくつかの点において欠点があるために,これらを克服するために考えられたのがこのHDLC手順です.IBM社のSDLCが元になっています.

HDLC方式は,フレーム・フォーマットから構成され,フレームごとに誤り制御を行うために,信頼性の高い伝送ができます.また,どのようなビット・パターンのデータでも送ることができ,透過性は完全に保証されています.

フレームは,フラグ・シーケンス,アドレス情報,制御,情報,FCSおよびフラグ・シーケンスから成っています(図10).

同期の確立は,フラグ・シーケンスによって行われ,誤り制御用のFCS(フレーム・チェック・シーケンス)フィールドは,16ビットのCRCで行われます.

これによって,HDLCは,高速で信頼性の高い伝送ができ,LANなどのコンピュータ・ネットワークに最適です.具体的なプログラム例はZ80 SIOの解説のところで行います.

用語解説

● **CCITT V.24, V.28**

CCITT V.24勧告は,「データ端末装置(DTE)とデータ回線終端装置(DCE)間の相互接続回路の定義」という課題で,同期,非同期通信,専用回線でのデータ通信,交換網でのデータ通信に使用する相互接続回路,動作条件について定めています.

CCITT V.28勧告は,「不平衡復流相互回路の電気的特性」で,相互接続回路の電気的特性について規定されています.

これについては,「モデムと電話網によるデータ通信,CCITT Vシリーズ勧告解説,林高雄編著,CQ出版社」に詳しく解説されています.

● **伝送制御手順(通信プロトコル)**

データ伝送を行ううえでの同期の確立,データの形式,誤り制御などを決めることです.

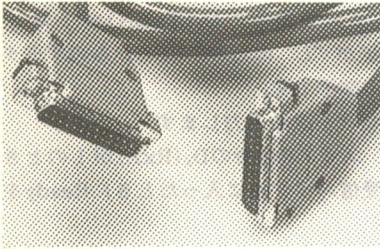
● **CRCとBCC**

CRC(Cyclic Redundancy Check)は,同期式伝送でよく利用される誤り検出方法のひとつです.

この方式は,送信するデータを数値(メッセージ多項式と呼ぶ)として,あらかじめ決められた値(生成多項式と呼ぶ)で割ります.その余りがBCC(Block Check Character)となります.

● **半2重と全2重について**

モデムなどのシリアル・インターフェース装置の仕様で,通信方式が半2重とか全2重といわれますが,双方向で伝送を行う場合の方式のことです.半2重は,どちらか一方のみが伝送でき,同時にお互いが伝送することはできません.全2重は,同時に伝送することができます.

半2重で問題となるのは,一方が伝送中であるとき,もう一方が伝送を中断するには,相手が伝送を終わるまで待つことになります.全2重で,伝送を中断したいときは,相手に割り込むことにより可能となります.

§2-2 シリアル・インターフェース用LSIの使い方

里 和政/相良富美/斉藤健司

シリアル・インターフェース用のLSIには，8251A，Z80 SIO，6850などがあります．表1に80系LSIの比較を示します．

8251A（USART）

8080/8086系のデータ・バス8ビット用シリアル・インターフェースLSIで，最も利用の多いICです．

◈ 8251Aの端子の機能

8251Aは，図1で示されるような28ピン構成で，図2のブロック・ダイヤグラムで示される以下のような機能をもっています．

(1) データ・バスとリード/ライト制御
(2) モデム制御
(3) 送信バッファと送信制御
(4) 受信バッファと受信制御

以上のような構成となっています．

端子機能を次に示します．

▶D_0〜D_7　システム側のデータ・バス信号線と接続する端子です．

▶$C/\overline{D}$　8251Aのデータ・バス上の情報がデータかコントロール・ワードまたはステータス情報であるかを区別し，“H”のときコントロール，“L”のときデータを示します．

▶$\overline{RD}$　この入力信号線が“L”のとき，8251Aからデータやステータス情報を読み出すことを示します．

▶$\overline{WR}$　CPUから8251Aにコントロール・ワードや，データを書き込む信号で，“L”入力アクティブです．

▶$\overline{CS}$　この入力信号が“L”のとき，8251Aをイネー

〈表1〉
各通信LSIの比較（80系）

比較項目＼種類	8251A	Z80 SIO	μPD7201
ピン数	28	40	40
チャネル数	1	2	2
データ・レート	〜9600bps（非同期） 〜56kbps（同　期）	〜55kbps（非同期） 〜880kbps（同　期）	〜55kbps（非同期） 〜880kbps（同　期）
非同期モード キャラクタ長 クロック ストップ・ビット パリティ・ビット 受信バッファ数	 5〜8 ×1，×16，×64 1，1.5，2 Even，Odd，なし 1	 5〜8 ×1，×16，×32，×64 1，1.5，2 Even，Odd，なし 3	 5〜8 ×1，×16，×32，×64 1，1.5，2 Even，Odd，なし 4
同期モード 同期 CRC 同期キャラクタ	 内部，外部 なし 1 or 2	 内部，外部 あり 1 or 2	 内部，外部 あり 1 or 2
HDLCモード CRC NRZI		 あり なし	 あり なし
割り込みモード	TxRDY RxRDY （Z80 モード1）	Z80 モード2	8085A モード 8086モード Z80 モード1
DMA機能	なし	なし	あり（制限付き）
内部レジスタ	2（パラメータ） 1（ステータス）	8（パラメータ） 3（ステータス）	8（パラメータ） 3（ステータス）
メーカ名	インテル，日本電気，沖電気，三菱，東芝など	ザイログ，シャープ東芝，ロームなど	日本電気など

ブルとするデバイス・セレクト信号です．

▶CLK　同期式の場合$\overline{TxC}$または$\overline{RxC}$入力の30倍以上，非同期式の場合は$\overline{TxC}$，$\overline{RxC}$入力が16×，64×の4.5倍以上のクロックを入力します．

▶RESET　8251Aを"idle"モードとし，最小6クロック分のパルス幅が必要です．

▶$\overline{TxC}$，$\overline{RxC}$　送受信キャラクタのボーレートを制御するクロックで，同期式の場合には実際のボーレートに等しく，非同期式の場合ボーレートの1倍，16倍，64倍のいずれかの周波数を入力します．

▶TxD　コマンド命令のD_0(TxEN)が1のときで$\overline{CTS}$入力端子が"L"のとき，CPUからのデータが出力される出力端子です．

▶RxD　受信データを入力します．

▶RxRDY　コマンド命令のD_2(RxE)が1のときでキャラクタが受信バッファに入ったときにReadyを表示

〈図1〉[7] 8251Aのピンの構成

種類	ピン番号	ピン名	信号名	I/O
データ・バス	1，2 5～8 27，28	D_0～D_7	データ・バス	I/O
	21	RESET	リセット	I
	20	CLK	クロック	I
	10	$\overline{WR}$	Write Data or Control	I
	13	$\overline{RD}$	Read Data	I
	12	C/$\overline{D}$	Control/Data	I
	11	$\overline{CS}$	Chip Enable	I
モデム・コントロール	22	$\overline{DSR}$	Data Set Ready	I
	24	$\overline{DTR}$	DataTerminal Ready	O
	17	$\overline{CTS}$	Clear to Send Data	I
	23	$\overline{RTS}$	Request to Send Data	O

種類	ピン番号	ピン名	信号名	I/O
トランスミッタ	19	TxD	Transmit Data	O
	15	TxRDY	Transmitter Ready	O
	18	TxE	Transmitter Empty	O
	9	$\overline{TxC}$	Transmitter Clock	I
レシーバ	3	RxD	Receive Data	I
	14	RxRDY	Receiver Ready	O
	25	$\overline{RxC}$	Receiver Clock	I
	16	SYNDET /BRKDET	Sync Detect /Break Detect	I/O
	26	Vcc	+5V	—
	4	GND	グラウンド	—

〈図2〉[1,7]
8251Aの内部ブロック図

データ・バス ㉗ ㉘ ① ② ⑤ ⑥ ⑦ ⑧ D0～D7
データ・バス・バッファ
制御ロジック
モデム制御
送信バッファ(並列→直列)
送信制御
受信バッファ(直列→並列)
受信制御
内部バス
システムのリセットに接続．リセットされた後は，コマンド・ポートにモード設定のフォーマットを書き込む　㉑ RESET
システムのクロックに接続　⑳ CLK
コマンド，データの選択．A8に接続　⑫ C/D
システム・バスの$\overline{IOR}$に接続　⑬ $\overline{RD}$
システム・バス$\overline{IOW}$に接続　⑩ $\overline{WR}$
アドレス・デコーダに接続．"L"のとき，8251Aがイネーブル　⑪ $\overline{CS}$
$\overline{DSR}$ ㉒ モデムの状態を検出する．ON：送信可，OFF：不可
$\overline{DTR}$ ㉔ データ・ターミナル・レディ
$\overline{CTS}$ ⑰ モデムが送信可を検出する，モデムのClear to Send
$\overline{RTS}$ ㉓ Request to Send
TxD ⑲ 直列変換された出力データ
TxRDY ⑮ データの転送がレディになったことを表す
TxEMPTY ⑱ トランスミッタのバッファが空になった
$\overline{TxC}$ ⑨ 変換の速度を決めるタイミング信号．通常は，送信，受信ボーレートが同一である
RxD ③ 直列入力データ．並列変換され，受信バッファへ蓄えられる
RxRDY ⑭ データが受信バッファに入ったときのレディ表示
$\overline{RxC}$ ㉕ 変換の速度を決めるタイミング信号
SYNDET ⑯ 同期式モードのみ使用する

し，割り込み信号などに使用します．

▶TxRDY　データの転送がReadyになったことを表し，この信号線は，データ・バッファが空で，TxENが1でかつ$\overline{\mathrm{CTS}}$が0のときアクティブとなります．

▶$\overline{\mathrm{DSR}}$　モデムの状態をテストするData Set Readyとして使用します．

▶$\overline{\mathrm{DTR}}$　モデムのData Terminal ReadyまたはRate Selectとして使用します．

▶$\overline{\mathrm{RTS}}$　モデムのRequest to Send dataとして使用します．

▶$\overline{\mathrm{CTS}}$　モデムのClear To Sendとして使用します．

▶TxE　"H" のとき，送信バッファから送信キャラクタがなくなったことを示します．

▶SYNDET/BD　この端子は，同期式モードのときに使用します．この信号線が，モード設定のときに出力として使用されると，受信モードの場合には，あらかじめ定められたSYNCキャラクタを受信すると "H" となります．

入力として使用すると，8251Aは，データ・キャラクタを，次の$\overline{\mathrm{RxC}}$の立ち下がりでアセンブルを開始します．

また，出力に使用した場合には，モード設定によって定められたビット数のデータが，すべてゼロの状態を検出したとき "H" となります．

▶V_{cc}　＋5Vを接続します．

▶GND　グラウンドを接続します．

● 送受信を行うとき気をつけなければならない動作

CPUによって8251Aにセットされたパラレル・データは，モード命令で指定したシリアル・データに変換され，TxDの出力端子から送り出されます．

ただし，送信データが送り出される条件は，コマンド命令で送信イネーブルとなっており，$\overline{\mathrm{CTS}}$が "L" のときです．また，最初の送信データが送り出されるまで，この端子はマーク状態になっています．

受信バッファは，RxD端子からシリアル・データを受信し，パラレル・データに変換してから，ステータスのRxRDYビットをセットします．

このときフレーミングとパリティのチェック(パリティがイネーブルのとき)を行い，エラーがあるとステータスにセットします．受信データをCPUに入力するときは，ステータスを入力してRxRDYがセットされていることを確認して実行します．

受信データをCPUに入力すると，RxRDYはリセットされますが，エラーが発生した場合には，コマンド命令を実行してリセットします．

◈ 8251Aのプログラミング

8251Aの設定は，まず図3で示すモード・レジスタによって動作モードを決めます．非同期モードには，伝送速度ファクタ，伝送キャラクタ長，パリティの有無およびストップ・ビット数(1，1.5，2ビットのいずれか)があります．

● 同期モードの設定

図4は，同期モードでのモード・レジスタです．SCSはSYNコードの数，ESDはSYNコードの検出を外部で行うか，8251Aで行うかを選択します．

伝送速度は，TxC，RxCに供給したクロックを分周して決められます．その分周比が伝送速度ファクタです．19.2kHzをクロックとした場合，1×で19200ボー，16×で1200ボー，64×で300ボーとなります．

通常は，16×がよく使用されています．ただしこれは，非同期モードで変えることができ，同期モードのときは1×の固定です．図5は，8251Aの動作フローチャートです．

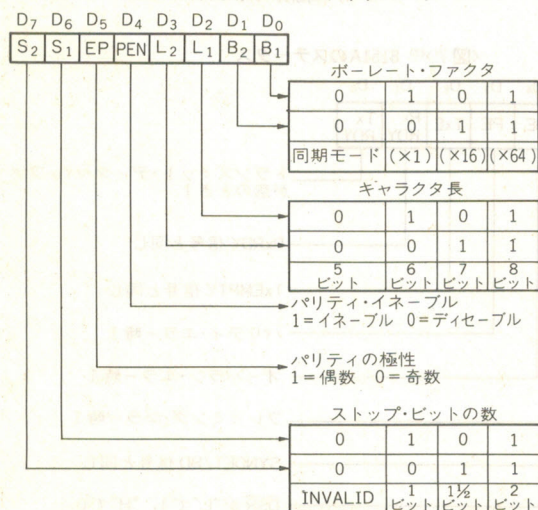
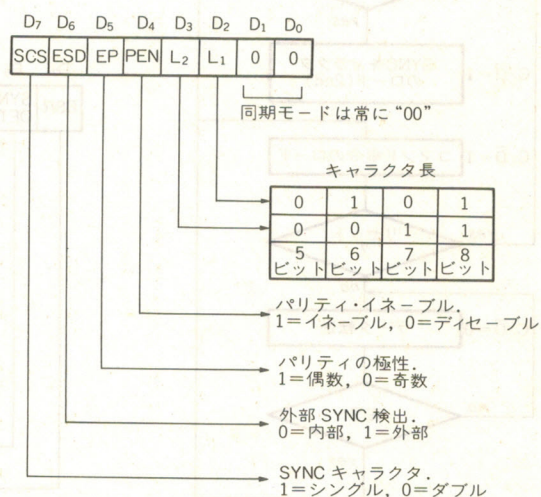
〈図3〉[7] 8251Aの非同期でのモード命令のフォーマット

〈図4〉[7] 8251Aの同期式でのモード命令フォーマット

● **コマンド・レジスタの設定**

図6にコマンド・レジスタを示します．コマンドによって伝送制御を行います．TxEは送信を行うときに“1”，RxEは受信を行うときに“1”にします．SBRKは，TxDを強制的に“L”にします．

ERは，受信エラー(PE, OE, FE)をリセットします．IRは，内部リセットしてモードを再設定にします．

EHは，ハント・モードに入るときに“1”にします．ハント・モードとは，同期モードにおいてSYNコードのみを受信することをいいます．

図7にステータス・レジスタを示します．ステータス・レジスタは，送信/受信の状態を確認します．TxRDYとTxEによって送信バッファが空になると“1”になり，TxEは送信シフト・バッファが空になると“1”になります．

PE, OE, FEは，受信エラーの状態を表します．PEは，パリティ・エラーがあったときにセットされます．OEはオーバラン・エラーと呼び，受信バッファにデータがあり，続けて受信して前のデータが破壊されたときにセットされます．FEはフレーミング・エラーと呼び，ストップ・ビットが正常に認識されないときセットされます．

SYNDETは，同期モードではSYNコードを検出したときにセットされ，非同期モードでは，ブレーク・キャラクタを検出したときにセットされます．

以上のレジスタにより伝送します(リスト1参照)．

注意点としては，RESET動作で6クロックかかり，モード・コマンドでは，非同期の場合8クロック，同期の場合16クロック分ICの内部処理がかかります(1クロック 最大3.125MHz)．

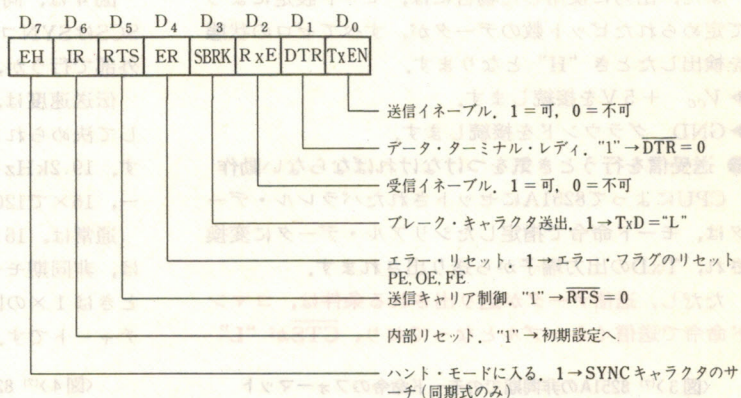
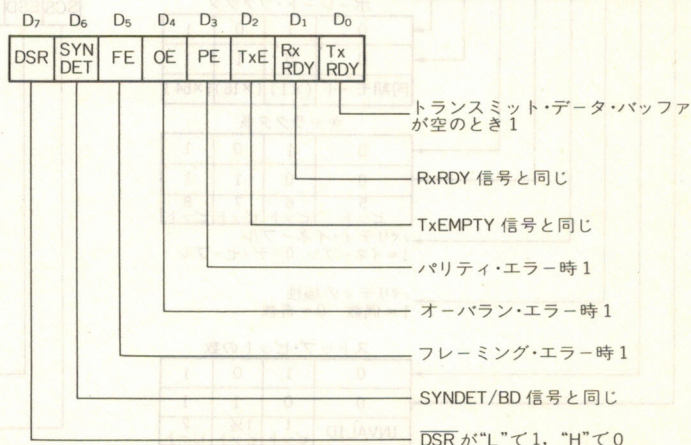
そのため，高速のCPUを使うときには，ウェイトが必要になります．

8251Aは，外部から送受信クロックを入力するために，そのクロックを8253(プログラマブル・タイマ)などを使用すると容易に，ボーレートを変更することができます．

6850/6350(ACIA)

ACIAは，その名のとおり調歩同期式直列インター

〈図5〉[1],[7] 8251Aの初期設定フローチャート

〈図6〉[7] コマンド・レジスタ

〈図7〉[7] 8151Aのステータス

〈リスト1〉
8251A による
シリアル入出力
(Lattice C)

```
/*
   USART-8251A  シリアル伝送(非同期)  プログラム
*/

#define csreg 0x00      /*  8251A  コントロール/ステータス  */
#define dtreg 0x01      /*  8251A  データ  ポート  */

#define reset 0x40      /*  リセット  コマンド  */
#define mode 0x04e      /*  モード設定  コマンド  */
                        /*  クロック分周比  1/16  */
                        /*  データ・ビット長  8ビット  */
                        /*  ストップ・ビット  1ビット  */
                        /*  パリティ  なし  */
#define rvcmd 0x6       /*  受信  コマンド  */
                        /*  受信イネーブル  */
                        /*  DTR  オン  */
#define sdcmd 0x23      /*  送信  コマンド  */
                        /*  送信イネーブル  */
                        /*  DTR  オン  */
                        /*  RTS  オン  */
#define eroff 0x10      /*  エラー  リセット  */
#define rvsts 0x03a     /*  レシーブ・レディまたはエラー  */
#define ersts 0x038     /*  レシーブ・エラー  */
#define sdsts 0x01      /*  センド・レディ  */
#define dsr 0x080       /*  送受信可能  */
/*  初期化  ルーチン  */

sinit()
{
        int i;
        for(i=0;i!=4;i++)               /*  ダミー  コマンド出力  */
        {
           outp(csreg,0x83);
        }
        outp(csreg,reset);              /*  コントローラ  リセット  */
        outp(csreg,mode);               /*  コントローラ  モード設定  */
        return(0);
}

/*  データ  受信  1バイト  */

data_in(data)
char data;
{
        int stat;

        outp(csreg,rvcmd);              /*  受信モード設定  */
        /*  データ受信まで待つ  */
        while((stat=inp(csreg)&rvsts)==0){}

        if((stat&ersts)!=0)
        {
          outp(csreg,rvcmd+eroff);              /*  エラー・ビット  リセット  */
          return(-1);                           /*  エラー  終了  */
        }

        data = inp(dtreg);                      /*  データ受信  */
        return(0);
}

/*  データ  バイト  送信  */

data_out(data)
char data;
{
        int stat;

        outp(csreg,sdcmd);              /*  送信モード設定  */
        while(((stat=inp(csreg))&sdsts)==0)     /*  ステータス  リード  */
        {
          if((stat&dsr)==0)
          {
                /*  送信不可  */
                return(-1);
          }
        }

        outp(dtreg,data);                       /*  データ  送信  */
        return(0);
}
```

フェースですから，8251Aのように同期式との切り替えはありません．

もともとBell103．600bpsモデムであるMC6860と組んで，電話回線で長距離データ通信ができるように設計されています．

図8にピン配置を示します．

◈ MPUがアクセスできるACIAのレジスタ

ACIAの内部には，MPUがアクセスできる4本のレジスタがあります．2本ずつ同じ内部アドレスに配置されています．図9にACIAの内部ブロック図を示します．

受信データ・レジスタとステータス・レジスタは読み出し専用ですし，送信データ・レジスタとコントロール・レジスタは書き込み専用になっていて，ACIAが一度に必要とするアドレスは2バイトだけです．

● コントロール・レジスタで付加ビットを設定する

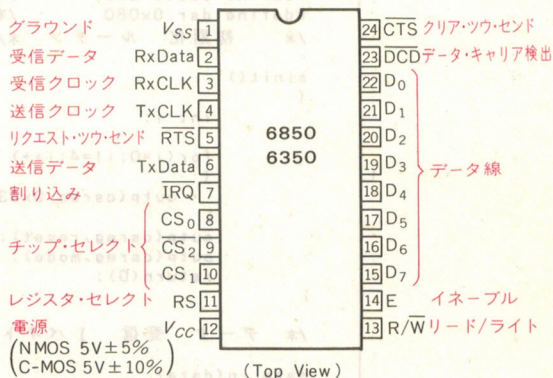
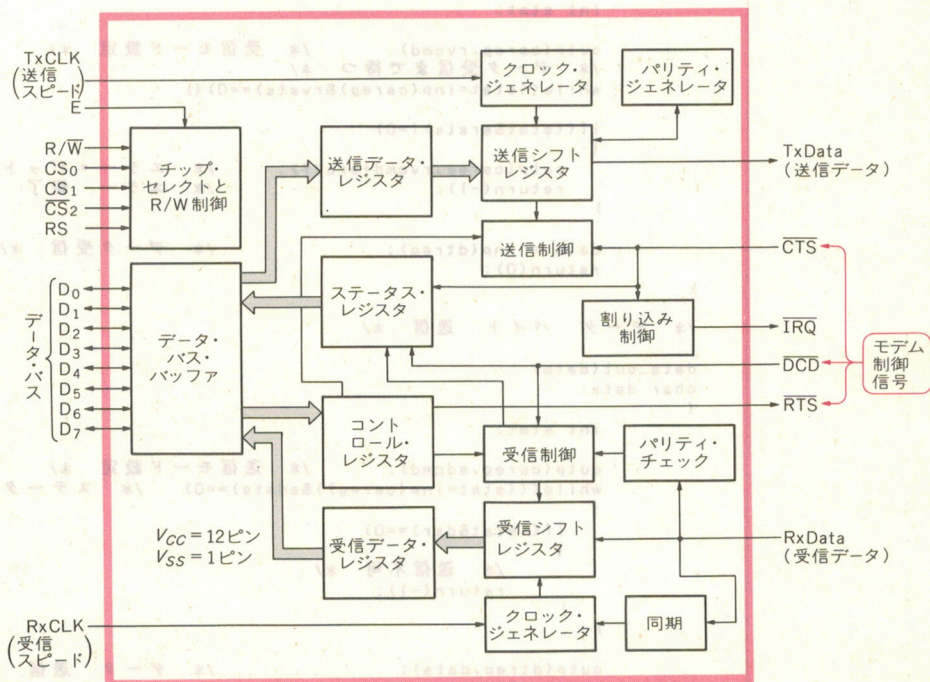
直列伝送を行うためには並列で扱っているデータを直列に，またその逆の変換をしなければなりません．簡単に思い付くのはシフトレジスタですが，実際にACIAにはシフトレジスタが入っているのです．ただたんに，並列-直列変換をしてもどこが始まりで終わりなのか受信側はわかりません．ですから送信側では，直列データにスタート，ストップ・ビットを付加してやります．

データの伝送は長距離を想定しているので，雑音が入らないとも限りません．そこで受信データの誤りを識別できるようにパリティ・ビットも付けられるようにします．パリティ・チェックですから，CRCのように正確ではありません．これらの付加ビットは，コントロール・レジスタの設定によって行います．

● 転送速度の決め方

並列→直列変換には，一定のクロック・パルスが必要です．周知のようにデータ通信では，転送速度を決めるボーレートというのがあり，これを設定するためには，シフトレジスタのクロック・パルスに相当するRxCLKとTxCLKが必要です．

ACIAでもソフトウェアでボーレートを変化させられるように，コントロール・レジスタの設定によって入力クロックを3段階に分周できます．なお，ACIAが扱える最大のボーレートは，Bバージョンで1000 kbpsまでです．

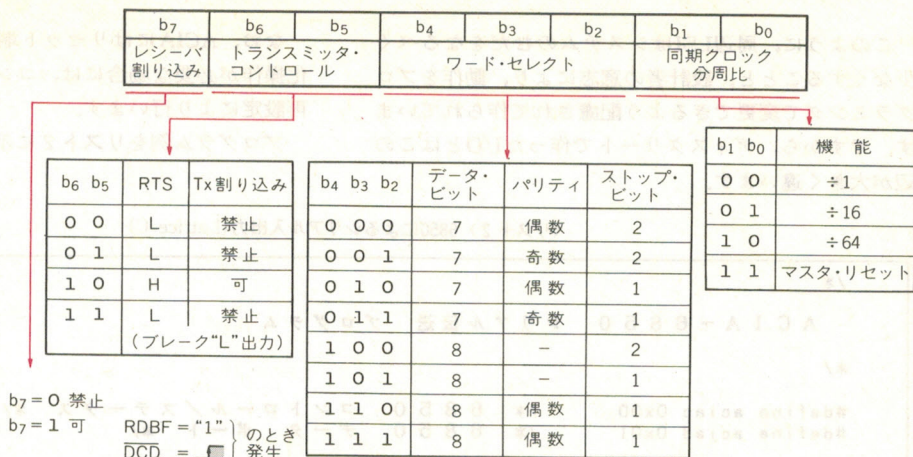
〈図8〉[10] 6850(ACIA)のピン配置

〈図9〉[10] 6850(ACIA)の内部ブロック図

〈図10〉[10]
6850(ACIA)のコントロール・レジスタ

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
レシーバ割り込み	トランスミッタ・コントロール		ワード・セレクト			同期クロック分周比	

b_6 b_5	RTS	Tx割り込み
0 0	L	禁止
0 1	L	禁止
1 0	H	可
1 1	L	禁止（ブレーク"L"出力）

b_4 b_3 b_2	データ・ビット	パリティ	ストップ・ビット
0 0 0	7	偶数	2
0 0 1	7	奇数	2
0 1 0	7	偶数	1
0 1 1	7	奇数	1
1 0 0	8	−	2
1 0 1	8	−	1
1 1 0	8	偶数	1
1 1 1	8	偶数	1

b_1 b_0	機 能
0 0	÷1
0 1	÷16
1 0	÷64
1 1	マスタ・リセット

b_7＝0 禁止
b_7＝1 可　RDBF＝"1"、$\overline{DCD}$＝立ち上がり のとき発生

● ステータス・レジスタはシフトレジスタの状態を示す

さて，直列伝送ではビット・データを1個ずつ送りますし，その転送速度によって，システム・バス内でやりとりされるスピードに比べてかなり遅くなってしまいます．したがって，前に送ったデータがACIA内のシフトレジスタに残っていれば，次のデータは書き込めません．

受信データも完全にシフトレジスタに入ってしまわなければ読み出すことはできません．このために，ACIAでもステータス・レジスタでこの情報を提供しています．

◈ ACIAのデータ送信/受信のシーケンス

図10にACIAのコントロール・レジスタの内容を示します．8251Aのようにデータ・ビット長にバリエーションはありませんし，ストップ・ビットの設定に1.5という半端なものもありませんので，RTTYなどの特殊な用途には使えないことになります．また，図11にステータス・レジスタの内容を示します．

その他に，通信用のLSIの機能として必要なものとして，モデム制御があります．つまり，回線からデータが入力されたときはいつでもMPUはこれに対応しなければなりませんが，そこでモデムは回線のキャリアを検出しACIAに知らせます．

ACIAはこれを受けてステータス・レジスタの$\overline{DCD}$フラグをセットし，もしIRQイネーブルであればMPUに対し割り込みをかけます．また，回線にデータを出力する場合には，モデムにこの要求を出します．回線が使用可能ならモデムは許可信号をACIAに出します．これでACIAは直列データを出力します．

◈ ACIAの割り込み

さて，ACIAの割り込みについて見てみます．直列データ転送は，一般に低速でありMPUがこのスピードで処理したのでは，システム全体の処理能力がたいへん低くなってしまいます．

〈図11〉[10] 6850(ACIA)のステータス・レジスタ

また，いつ入ってくるのか予測のつかないデータをいつまでも待つわけにもいきません．ACIAの割り込み要因を列挙してみます．

- 送信レジスタが空で，モデムからの送信許可があるとき
- モデムが受信キャリアを検出したとき
- 受信レジスタにデータがきちんと入ったとき
- 受信データがオーバランしたとき

これらは，もちろんコントロール・レジスタの割り込みイネーブル・ビットがセットされていなければ，ハード的な割り込みは発生せず，ステータス・レジスタに結果が残るだけになります．

このように，周辺LSIはシステムのむだをなるべく少なくすることと，設計者の意志により，動作をプログラミングで変更できるよう配慮されて作られています．ですから，ディスクリートで作ったI/Oとはこの辺が大きく違います．

なお，ACIAにはリセット端子がありません．初期化操作が必要な場合には，コントロール・レジスタの再設定により行います．

プログラム例をリスト2に示します．

〈リスト2〉6850によるシリアル入出力(Lattice C)

```
/*

  ACIA-6850  シリアル伝送  プログラム

*/

#define aciac 0x00      /*  6850  コントロール／ステータス  */
#define aciad 0x01      /*  6850  データ  ポート  */

#define reset 0x03        /*  6850  リセット  コマンド  */
#define mode 0x015        /*  6850  モード設定  コマンド  */
                          /*  クロック分周比  1/16  */
                          /*  データ・ビット長  8ビット  */
                          /*  ストップ・ビット  1ビット  */

#define rvsts 0x071       /*  レシーブ・レディまたはエラー  */
#define ersts 0x070       /*  レシーブ・エラー  */
#define sdsts 0x01        /*  センド・レディ  */
#define cts 0x08          /*  センド可能  */

/*  6850  初期化  ルーチン  */

sinit()
{
        outp(aciac,reset);        /*  コントローラ  リセット  */
        outp(aciac,mode);         /*  コントローラ  モード設定  */
        return(0);
}

/*  データ  受信  1バイト  */

data_in(data)
char data;
{
        int stat;

        /*  データ受信まで待つ  */
        while((stat=inp(aciac)&rvsts)==0){}

        if(stat==ersts)
        {
           inp(aciad);        /*  エラー・ビット  リセット  */
           return(-1);        /*  エラー  終了  */
        }
        data = inp(aciad);      /*  データ受信  */
        return(0);
}

/*  データ  バイト  送信  */

data_out(data)
char data;
{
        int stat;

        while(((stat=inp(aciac))&cts+sdsts)==0) {}  /*  ステータス  リード  */

        if(stat==cts)                    /*  送信可能か？  */
           return(-1);                   /*  送信不可  */
        outp(aciad,data);                /*  データ  送信  */
        return(0);
}
```

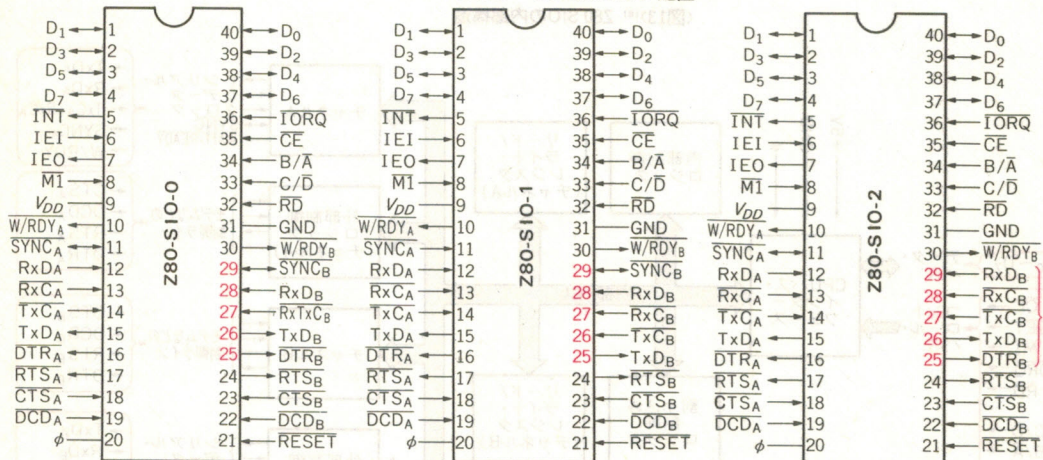
Z80 SIO

Z80 SIOは，Z80用のLSIとして設計され，非同期，同期，SDLC(HDLC)通信を制御できるLSIです．伝送用のチャネルは二つです．

● Z80 SIOの特徴

8251Aでは，同期式の場合CRCの計算などを，ソフトウェアによって行う必要がありますが，このLSIはこれらの機能が内蔵されています．

伝送速度は，Z80Aを4 MHzクロックとしたとき800kビット/秒です．

〈図12〉(8) Z80 SIOのピン配置

〈表2〉(8) Z80 SIO信号の説明

端子名	I/O	機能
D_0～D_7	双方向	Z80 CPUのデータ・バスと接続し，CPUとSIOの間のデータ転送に使用する
$B/\overline{A}$	入力	SIOのAまたはBチャネルを選択．通常，CPUのアドレス線A_1を接続する
$C/\overline{D}$	入力	この信号は，CPUとSIO間で転送される信号が，制御コマンドであるかデータであるかを区別する．通常，CPUのアドレス線A_0を接続する
$\overline{CE}$	入力	チップ・イネーブル信号でアドレスをデコードした信号を接続する
ϕ	入力	内部信号の同期をとるために使用．システム・クロックを接続する
$\overline{M1}$	入力	CPUの$\overline{M1}$信号を接続する．$\overline{IORQ}$信号とともにインターラプト・アクノレッジ信号として使用したり，RETI命令を解釈するのに用いる
$\overline{IORQ}$	入力	CPUのIORQ信号に接続する．CPUとSIO間でデータを転送する場合，$B/\overline{A}$，$C/\overline{D}$，$\overline{CD}$，および$\overline{RD}$と組み合わせて使用する
$\overline{RD}$	入力	CPUの$\overline{RD}$信号と接続する．$\overline{CE}$とIORQがアクティブで，かつ$\overline{RD}$もアクティブであればSIOからデータを読み出す．CEとIORQがアクティブで，$\overline{RD}$が非アクティブであればSIOにデータを書き込む
$\overline{RESET}$	入力	SIOをリセット
IEI	入力	割り込みの優先順位を決めるデイジィ・チェーン回路をIEOと共に構成する．この信号が“H”の時，SIOは割り込みを行うことができる
IEO	出力	IEIと共にデイジィ・チェーン回路を構成する．IEOが“L”の場合，そのデバイスあるいはそれより優先順位の高いデバイスが割り込みを実行中であることを示す
$\overline{INT}$	出力	SIOが割り込み要求をすると“L”になる
$\overline{W/RDY_A}$ $\overline{W/RDY_B}$	出力	DMAコントロール用レディ線，またはCPUとSIO間のデータ転送速度を合わせるためのウェイト線として利用
$\overline{CTS_A}$ $\overline{CTS_B}$	入力	オート・イネーブルにSIOがプログラムされている場合，この信号をアクティブにすると対応するトランスミッタがイネーブルになる
$\overline{DCD_A}$ $\overline{DCD_B}$	入力	オート・イネーブルにSIOがプログラムされている場合，この信号をアクティブにすると対応するレシーバがイネーブルになる
RxD_A RxD_B	入力	受信データ線
TxD_A TxD_B	出力	送信データ線
RxC_A RxC_B	入力	受信クロックで，受信データは$\overline{RxC}$の立ち上がりでサンプリングされる
$\overline{TxC_A}$ $\overline{TxC_B}$	入力	送信クロックで，送信データは$\overline{TxC}$の立ち下がりエッジで変化する
$\overline{RTS_A}$ $\overline{RTS_B}$	出力	書き込みレジスタ内のRTSビットをセットするとアクティブになる．SIOが送信したいときにRTSビットをセットする
$\overline{DTR_A}$ $\overline{DTR_B}$	出力	書き込みレジスタ内のDTRビットをセットするとアクティブになる．SIOが受信できる状態になればこの信号をアクティブにする
$\overline{SYNC_A}$ $\overline{SYNC_B}$	双方向	外部同期通信に使用する場合，外部同期が成立したらこの信号をアクティブする

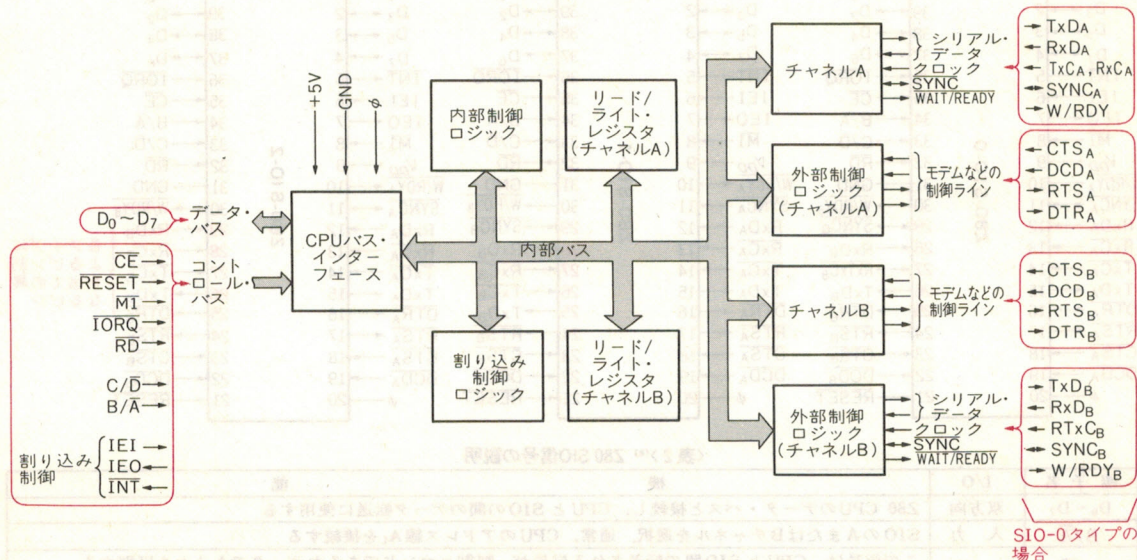
図12にZ80 SIOのピン配置を示しますが，チャネルAにはすべての信号があります．表2に信号の説明を，図13に内部構造を示します．

◈ Z80 SIOのレジスタの設定

図14に内部レジスタを示します．レジスタの設定は，まずWR_0で設定するレジスタを選択したうえでレジスタに書き込みます．WR_0のときは，1回です．

このとき，$B/\overline{A}$信号で設定するチャネルを選択し，$C/\overline{D}$信号は“1”にします．

WR_0は，レジスタの選択とコントローラの動作を指定します．WR_1は，割り込みコントロール・レジスタで，割り込みを使用しない場合は，すべて“0”にします．WR_2もWR_1と同様で割り込み時のジャン

〈図13〉[8] Z80 SIOの内部構造

〈図14〉[11] Z80 SIOのレジスタ

書き込みレジスタ0：WR_0

(ビット：D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0)

D_2	D_1	D_0			A	B	S
0	0	0	レジスタ0	レジスタ・ポインタ	○	○	○
0	0	1	レジスタ1		○	○	○
0	1	0	レジスタ2		○	○	○
0	1	1	レジスタ3		○	○	○
1	0	0	レジスタ4		○	○	○
1	0	1	レジスタ5		○	○	○
1	1	0	レジスタ6		○	○	○
1	1	1	レジスタ7		○	○	○

D_5	D_4	D_3			A	B	S
0	0	0	動作に何の影響も与えない	制御コマンド	○	○	○
0	0	1	アボート送出		/	/	○
0	1	0	外部ステータス割り込みリセット		○	○	○
0	1	1	チャネル・リセット		○	○	○
1	0	0	つぎの受信キャラクタで割り込みイネーブル		○	○	○
1	0	1	送信割り込みの保留リセット		○	○	○
1	1	0	エラー・リセット		○	○	○
1	1	1	割り込みからの復帰(Aチャネルのみ)		○	○	○

D_7	D_6			A	B	S
0	0	動作に何の影響も与えない	CRCリセット・コード	○	○	○
0	1	受信CRCチェッカ・リセット		/	○	○
1	0	送信CRCジェネレータ・リセット		/	○	○
1	1	送信アンダラン/EOMリセット		/	○	○

書き込みレジスタ1：WR_1

(ビット：D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0)

ビット		A	B	S
D_0	外部/ステータス割り込みイネーブル	○	○	○
D_1	送信割り込みイネーブル	○	○	○
D_2	ステータス・エフェクト・ベクタ・イネーブル(Bチャネルのみ)	○	○	○

D_4	D_3			A	B	S
0	0	受信割り込みディセーブル	割り込みモード	○	○	○
0	1	最初のキャラクタのみで受信割り込み		○	○	○
1	0	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件となる)		○	○	/
1	1	すべての受信キャラクタで割り込み(パリティ・エラーは特別受信条件とならない)		○	○	○

D_7	D_6	D_5			A	B	S
0	0	×	$\overline{WAIT}$：フローティング	ウェイト・レディ機能選択	○	○	○
0	1	×	$\overline{READY}$：“H”レベル		○	○	○
1	0	0	$\overline{WAIT}$：送信バッファがフルかつSIOのデータ・ポートが選択されているとき“L”レベル ：送信バッファが空のときフローティング		○	○	○
1	1	0	$\overline{READY}$：送信バッファがフルのとき“H”レベル ：送信バッファが空のとき“L”レベル		○	○	○
1	0	1	$\overline{WAIT}$：受信バッファがフルのときフローティング ：受信バッファが空かつSIOのデータ・ポートが選択されているとき“L”レベル		○	○	○
1	1	1	$\overline{READY}$：受信バッファがフルのとき“L”レベル ：受信バッファが空のとき“H”レベル		○	○	○

(注) A：非同期，B：同期，S：SDLC の各通信方式にて，/：未使用(0にプログラム)，○：使用を意味する．

〈図14〉[11] Z80 SIOのレジスタ(つづき)

〈図15〉[5],[8] Z80SIOの回路図

プ・ベクトルを指定します．

WR_3は，8251Aのコマンドにあたり，受信の制御を指定します．WR_4は，8251Aのモードにあたりコントローラのモードを指定します．WR_5は，送信の制御を指定します．WR_6,WR_7は，同期モード，SDLCモード時に指定します．

ステータス・レジスタは，RR_0だけで直接読み出すことができます．RR_1,RR_2はWR_0でレジスタを指定します．

RR_0は，データの受信/送信状態がセットされます．RR_1は，エラーの状態とSDLCモードでの受信ビットの端数がセットされます．RR_2は，割り込み時のベクトルがセットされます．

回路例を図15に示します．回路図は，SIO-2のタイプを用い2チャネル使っています．DMAは使用しません．

◆ Z80 SIOのプログラミング

参考までにSDLCのプログラムをリスト3に示します．初期化ルーチンでは，コントローラをSDLCモードにして，送受信を可能にしています．

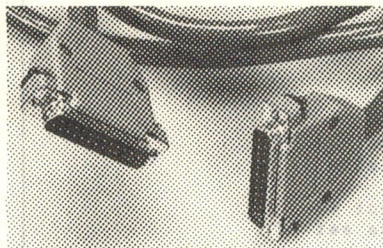
送信ルーチンではフレームを開始して，CRCの計算，フレーム終了時にCRCを送ります．受信ルーチンは，フレームを受信してCRCのチェックをします．

8080系用に日本電気製のμPD7201があります．基本機能は，Z80 SIOと同じです．このLSIについては「トランジスタ技術スペシャルNo.8」を参照してください．

〈リスト3〉 Z80 SIOによるシリアル入出力(Lattice C)

```
#define chadat 0x00    /*  チャネルA  データ      */
#define chacmd 0x01    /*  チャネルA  コマンド    */
#define chasts 0x01    /*  チャネルA  ステータス  */

#define reset  0x18    /*  リセット  コマンド  */
#define mmode  0x20    /*  SDLC  モード  */
#define rmode  0xcc    /*  受信  モード  */
                       /*  キャラクタ8ビット  */
                       /*  CRC許可、アドレス・サーチ  */
#define smode  0xe3    /*  送信  モード  */
                       /*  DTRオン、RTSオン、CRC許可  */
                       /*  キャラクタ8ビット  */
#define caddr  0x01    /*  チェック  アドレス  */
#define sflag  0x7e    /*  フラグ  シーケンス  */
#define rrdy   0x01    /*  受信レディ・ビット  */
#define srdy   0x04    /*  送信レディビット  */
#define err    0xf0    /*  エラー、エンド・ビット  */
```

〈リスト3〉Z80 SIOによるシリアル入出力(つづき)

```
/* コントローラ 初期化 */

sio_init()
{
        outp(chacmd,reset);          /* reset */

        outp(chacmd,0x01);           /* wr1 select */
        outp(chacmd,0x00);           /* intr disable */

        outp(chacmd,0x04);           /* wr4 select */
        outp(chacmd,mmode);          /* SDLC mode set */

        outp(chacmd,0x03);           /* wr3 select */
        outp(chacmd,rmode);          /* recive mode set */

        outp(chacmd,0x05);           /* wr5 select */
        outp(chacmd,smode);          /* send mode set */

        outp(chacmd,0x06);           /* wr6 select */
        outp(chacmd,caddr);          /* check address */

        outp(chacmd,0x07);           /* wr7 select */
        outp(chacmd,sflag);          /* flag bit */
        return(0);
}

/* 受信 ルーチン */
recv(data)
char data;
{
        int stat,er_cd;

        outp(chacmd,0x43);           /* wr3 select and crc reset */
        outp(chacmd,rmode+0x11);     /* auto hant mode */
        while(((stat=inp(chasts))&rrdy)!=rrdy)  /* 受信待ち */
        {
            if((stat&0x80)!=0)       /* アボート? */
            {
                return(-2);          /* アボート終了 */
            }
            outp(chacmd,0x01);       /* rr1 select */
            if(er_cd=(inp(chasts)&err)!=0) /* rr1 リード */
            {
                outp(chacmd,0x30);        /* エラー・リセット */
                return(er_cd);       /* エラー、エンド・フレーム */
            }
        }

        /* データ 受信 */

        data=inp(chadat);
        return(0);
}

/* データ 送信 */

send(data)
char data;
{

        outp(chacmd,0x85);           /* wr5 select and crc reset */
        outp(chacmd,smode+0x04);     /* 送信許可 */
        outp(chacmd,0xc0);           /* reset underrun */

        while((inp(chasts)&srdy)==0)      /* 送信レディ? */
        {}
        outp(chadat,data);                /* データ送信 */
        return(0);
}
```

§2-3

PC9801によるBSC伝送の実際

沖野 新

伝送制御手順

ここでは，BSC(バイシンク)によるデータ伝送を行う手順を，パソコンを用いて説明していきます．

BSCでは，図1のシーケンスにしたがって伝送を行います．つまり，送信側と受信側がそれぞれ相互に結ばれたことを確認し，その後テキストを送ります．そして，通信が終わったので，回線を切るといった手順を決めてあるわけです．

◈ 回線接続シーケンス

● 送信側

ターミナル呼び出しを行うためにENQを送信します．ターミナルが複数台あれば，IDにターミナル番号を付加します．その後，受信状態に入り応答受信待ちになります．

ACK_0を受信した場合は，正常に回線接続が行われたと見なし，テキストの送信を行います．

NAKを受信した場合は，再度ENQを送信します．

受信側の応答がない場合は，通常約2～5秒時間監視を行い，時間監視終了後(以下タイム・アウトと呼ぶ)，ENQの再送を行います．

WACK, ACK_1を受信した場合は，NAKと同様の処理を行います．

● 受信側

ENQ受信後，以後テキスト・ブロックを正常に受信できる状態であれば，ACK_0を送信します．正常に受信できる状態でなければ，WACKを送信します．

それ以外の制御コードであれば，応答は行いません．

伝送コードについて

図Aに，伝送コードを示します．各レコード先頭には，PAD_Lが1バイト，SYNが1または2バイトあります．これは，BSC手順が同期式で伝送されるからです．これによって，伝送キャラクタの同期を行います．

PAD_Lコードは，SYNコードより先行して，同期キャラクタの位置合わせに使用されます．

PAD_TはＦＦｈで，レコードの最後を示します．

● ENQコード

このコードは送信側から送られます．これによって，受信側はデータ受信の状態に入ります．IDの部分は，複数のターミナルの選択に使用されます．1台であれば不要です．

● STXコード

このコードはテキスト・レコードであることを示します．STXコードの次から，伝送するテキストを送ります．通常テキスト長は256バイトです．このレコードの後は，ETBまたはETXとBCC(Block Check Character)です．

ETBは，次に送るSTXコードがある場合に使用します．

ETXは，最後のSTXコードであることを示します．

BCCは，テキスト部のチェック・キャラクタです．これについては別に述べます(p.58参照)．

● EOTコード

このコードは送信側の終了を示します．これによって伝送を終結させます．

● ACK_0およびACK_1

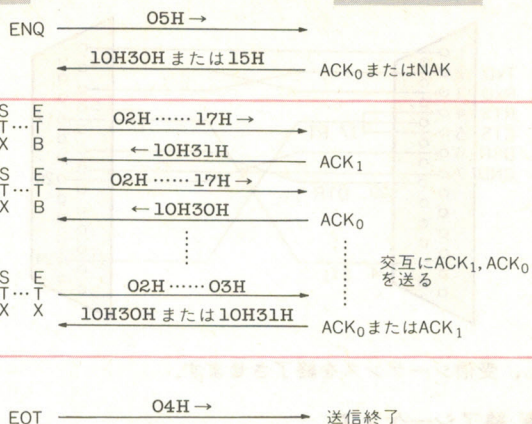
これは送信データを正常に受信したときに，受信側から送るコードです．

ENQコードを受信した場合は，ACK_0を送信します．以後，STXコード受信ごとに，交互にACK_1，ACK_0を送信します．

● NAKコード

このコードは，受信側が正常にデータを受信できないときに送信します．または，BCCエラー，シーケンス・エラーのときにも送ります．

送信側は，これによってテキスト・レコードの再送信，または再度ENQコードから始めます．

〈図1〉
伝送制御シーケンス

◈ テキスト伝送シーケンス

● 送信側

回線接続後，テキスト・ブロックの送信を行います．

ACK$_0$, ACK$_1$を受信した場合には，テキストがあればテキストを送信し，なければ終了信号のEOTを送信します．

テキスト・ブロックは，STXから始まり，ETB, ETXまでです．ETBは，これに連続するテキスト・ブロックがある場合に使用し，ない場合にはETXを使用します．

NAKを受信した場合は，直前のテキスト・ブロックを再度送信します．

タイム・アウトであれば，再度回線接続(ENQ)から始めます．

WACKを受信した場合は，ただちにENQの送信を行います．

● 受信側

テキスト・ブロックの受信を行います．

テキストを正常に受信し，BCCエラーがない場合に，ACK$_0$またはACK$_1$を送信します．

正常に受信できない，もしくはBCCエラーが発生した場合には，NAKを送信します．

ENQを受信した場合は，直前に送信したACK$_0$, ACK$_1$を再度送信します．

EOTを受信した場合は，データ伝送の終了と見な

● WACKコード

このコードは受信側が受信状態を一時中断する場合に使います．

● RVIコード

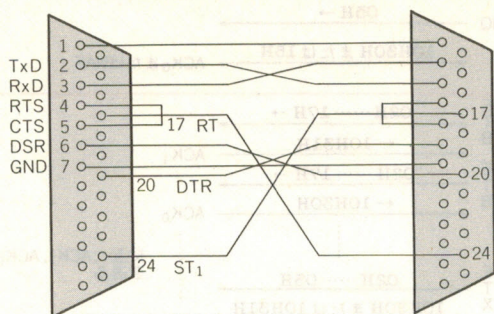
受信側が送信側になる場合に，このコードを送信側に送ります．

〈図A〉
伝送制御レコード・フォーマット

ENQコード	PAD$_L$	SYN	SYN	(ID)	ENQ	PAD$_T$			
ACK$_0$ または ACK$_1$	PAD$_L$	SYN	SYN	(ID)	ACK$_0$/ACK$_1$	PAD$_T$			
NAK	PAD$_L$	SYN	SYN	NAK	PAD$_T$				
STX + ETB/ETX	PAD$_L$	SYN	SYN	STX	テキスト	ETB/ETX	BCC	BCC	PAD$_T$
EOT	PAD$_L$	SYN	SYN	EOT	PAD$_T$				
WACK	PAD$_L$	SYN	SYN	WACK	PAD$_T$				
RVI	PAD$_L$	SYN	SYN	RVI	PAD$_T$				

〈図2〉同期式RS-232Cの接続

〈図3〉PC9801のディップ・スイッチの設定

スイッチ	
7	OFF
8	ON
9	ON
10	OFF

し，受信シーケンスを終了させます．

◈ 終了シーケンス

● 送信側

テキスト・ブロックをすべて送信した場合に，EOTを送信します．

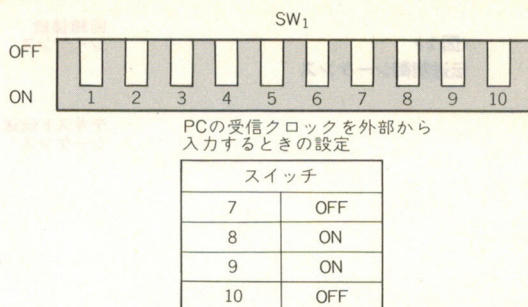
伝送中のタイム・アウト・エラーが発生した場合にも，EOTを送信します．

● 受信側

すべての受信処理を終了させます．

次に実際パソコン上で動くプログラムを示します．

ハードウェアの構成

最近のパーソナル・コンピュータは，ほとんどRS-232Cが標準となっていますので，容易にBSC伝送が行えます．

PC9801を用いて，BSC伝送を行った例をここで説明します．ほかのパーソナル・コンピュータも同様に使用できます．

PC9801(以下PCとする)は，RS-232C用のコントローラに8251Aを使用しています．

BSCでは，前にも述べたように，同期式で通信しなければなりませんが，PCでは非同期式が標準となっていますから，同期式に変更します．これは，直接コントローラに対して，モード設定を行うことでできます．

同時に，伝送速度クロックも変更する必要があります．PCでは8253を使っています．このプログラムでは4800bpsにしました．

BCC(Block check character)とは

伝送中にテキストのデータ抜けまたは，パリティでは発見不可能な二重パリティ・エラーなどを容易に発見するための2バイトのキャラクタです．

BSCの伝送では，CRC16またはCRC12が使用されています．

最近のコントローラでは，このCRCの計算ロジックを内蔵していますが，8251Aではソフトウェアによって実現できます．

BCCのチェック範囲は，STXコードの次のテキストからETBまたはETXまでです．

チェックの方法は，送信側ではBCCを計算し，その結果を送信します．受信側は，データの受信を行いながらBCCを計算して，1レコード受信後，BCCの比較を行います(図B参照)．

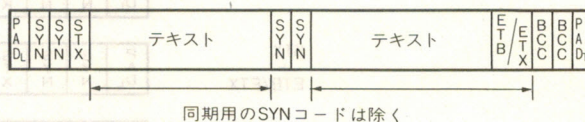
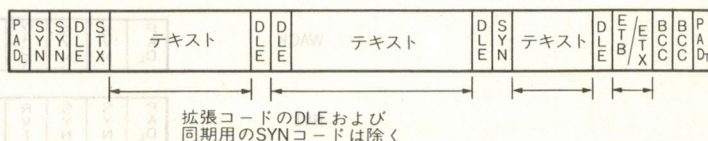
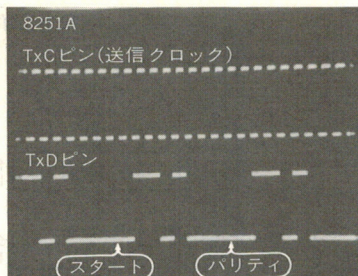
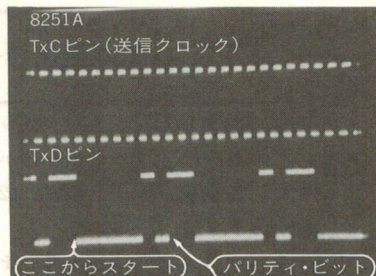
〈図B〉
BCCの計算範囲

〈写真1〉
SYNコードの送信
〔X；2 ms/div,
Y；2 V/div〕

〈写真2〉
"A"の送信
〔X；2 ms/div,
Y；2 V/div〕

RS-232Cの接続は，無手順と同様ですが，同期式の場合は，相手の送信クロックを受信クロックとしたほうがよいでしょう．

図2に接続方法，図3にPCでの送受信クロックの変更方法を示します．

参考として同期式伝送時の波形を**写真1**，**写真2**に示します．共に伝送速度は，4800bpsで8251AのTxDの状態です．

写真1は，SYNコード(16H)を送信したときの波形です．**写真2**は，"A"(41H，01000001B)を送信したときの波形です．

伝送プログラム

次に，パーソナル・コンピュータを使った，ファイル伝送について必要なことがらを説明します．

● **初期化ルーチン**

このルーチンでは，インターフェース・コントローラ8251Aおよび伝送速度の設定を行います．

モード設定は，同期モード，キャラクタ長は8ビット，奇数パリティ，同期キャラクタは2バイトでSYN(32H)とします．

● **受信メイン・ルーチン**

伝送レコードを受信して，各処理を行います．

● **送信メイン・ルーチン**

回線をオープンし，テキスト・ブロックを伝送します．

● **受信処理**

ハント・モードを設定後，ハント状態に入ります．ハントは，同期キャラクタを受信後，ステータス・フラグをセットすることによって終わります．同期確立後，キャラクタの受信を行います．レコードの終了は，PAD_Tを受信後です．

STXを受信した場合は，ETBまたはETXまでをテキストとして受信します．そしてこれに続くBCC 2バイトを受信しBCCの比較を行います．

受信バッファ長は，1テキスト長256バイトとして最大300バイト確保します．

BSC伝送のクロック

PC9801VMでは，8251Aのクロックの切り替えを変更しているために，受送信共にクロックを外部から入力しなければなりません．このため，各接続コネクタのST_1(24)ピンとST_2(15)ピンを接続する必要があります．

ディップ・スイッチは，5番目をONにします(図C参照)．

〈図C〉[9] PC9801のSW_1の5/6の機能

			スイッチ5	スイッチ6	
SW_1	5	RS-232Cの伝送速度（ボーレート）を決めるためのタイマ選択	ON	ON	BCI同期
			ON	OFF	ST_2同期
	6		OFF	ON	同期刻時機構
			OFF	OFF	調歩同期(非同期)

備　考　スイッチ5，6はRS-232Cの伝送速度を決めるためのタイマを選択する．
VMのSW_1の6は未使用．

スイッチ5	スイッチ6	機　能
ON	ON	BCI同期……送信用のタイミングとして，PC9801VXの内部タイマを使用する．受信用のタイミングはモデムより供給されるクロックを使用する．
ON	OFF	ST_2同期……送，受信用のタイミングとしてモデムより供給されるクロックを使用する．

〈図4〉 同期式初期化フローチャート

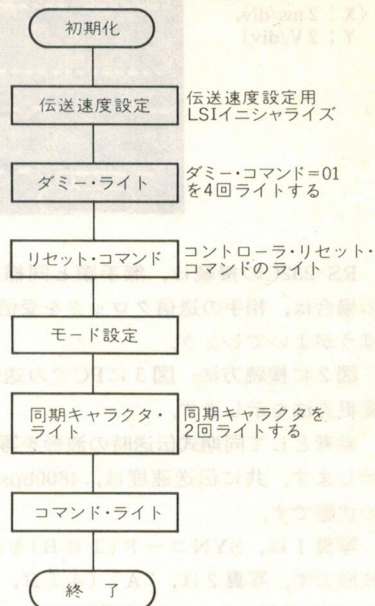
受信エラー時とタイム・アウトの場合は，すぐに受信を中断します．このプログラムでは，約2秒のタイム・アウトに設定しています．デバッグ時には，長くしたほうがよいと思われます．

● 送信処理

各伝送制御コードにPAD_LおよびSYNを付加して送信します．

テキストを送信する場合，あらかじめ，STX，BCC，ETBまたはETXを付加したレコードを用意しておき，これを送信します．

テキスト長は最大256バイトとします．

8251Aに対してI/Oの操作を行うルーチンは，無手順の場合と同様です．

これらのフローチャートを，図4～図7に示します．

プログラムを**リスト1**に示します．このプログラムは，MS-DOS上にて作成しました．

● おわりに

最後に，BSC手順は同期式のプロトコルですが，非同期式でも使用できます．この場合，同期キャラクタおよび同期を確立するハント・モードが不要となります．

しかし，スタート/ストップ・ビットが付加されますから，伝送効率が約20%ほど悪くなります．

また，割り込み処理を使用しない場合に，伝送中，ディスクなどをアクセスすると，受信側でタイム・アウトが発生することがあります．

〈図5〉受信メイン・フローチャート
受信メイン
レコード受信
ENQか？
no
yes
ACK₀を送信
SD.ACK＝1
次に送信するACK(ACK₀, ACK₁)
レコード受信
タイム・アウト
no
yes
EOT送信
ETB
yes
no
ETX
yes
no
ACK送信
SD.ACKに従う
EOT
yes
no
NAK送信
終　了
〈図6〉レコード受信フローチャート
レコード受信
受信モード設定
ハント・モードにし，受信可能にする
ステータス・リード
ハント中か
D₆＝1
yes
no
1キャラクタ・リード
SYN？
yes
no
STX？
no
yes
受信バッファにセット
1キャラクタ・リード
PAD_T
no
yes
伝送コードのセット
1キャラクタ・リード
ETB・ETX
yes
no
受信バッファにセット
BCCの計算
BCCリード
BCC比較
終　了
〈図7〉送信メイン・フローチャート
送信メイン
A
ENQ送信
レコード受信
ACK₀
no
yes
テキスト終了
yes
no
EOT送信
終　了
テキスト・レコード作成
テキスト送信
レコード受信
タイム・アウト
yes
no
NAK？
yes
no
ACK？
yes
no
A

〈リスト1〉PC9801によるBSC手順

```
;*************************
;*
;*  B S C  プ ロ グ ラ ム
;*
;*************************

code    segment byte

        assume  cs:code,ds:code,es:code

;
text_len        dw      ?
text_off        dw      ?
text_seg        dw      ?
text_addr       equ     dword ptr text_off

recv_len        dw      ?
send_len        dw      ?
recv_buff       db      300 dup(?)
send_buff       db      300 dup(?)

crc_cnt         dw      ?
crc_low         equ     byte ptr crc_cnt
crc_high        equ     byte ptr crc_low+1
ack_flag        db      ?
snd_ack         db      ?
send_flag       db      ?
                db      0
;
; test
;
start:
        mov     ax,cs
        mov     es,ax
        mov     ds,ax
        call    init            ; 8251 init
        call    send_entry      ;
        jmp     start


sio_stat        equ     032h    ;  8 2 5 1 A  ス テ ー タ ス
sio_cmd         equ     032h    ;  8 2 5 1 A  コ マ ン ド
sio_data        equ     030h    ;  8 2 5 1 A  デ ー タ
tim_ch2         equ     075h    ;  8 2 5 3    チ ャ ネ ル  2
tim_cmd         equ     077h    ;  8 2 5 3    コ マ ン ド

enq             equ     02dh
syn             equ     032h
stx             equ     002h
soh             equ     001h
etx             equ     003h
etb             equ     026h
nak             equ     03dh
dle             equ     010h
eot             equ     037h
wack            equ     06bh
ack0            equ     070h
ack1            equ     061h
padl            equ     055h
padt            equ     0ffh

;
; 8 2 5 1 A  初期化
;
init:
        mov     al,0b6h         ; チ ャ ネ ル  2  モ ー ド  3
        out     tim_cmd,al      ; タ イ マ  モ ー ド  セ ッ ト
        mov     al,0
        out     tim_ch2,al      ; L O W
        mov     al,2
        out     tim_ch2,al      ; H I G H
        mov     al,1
        out     sio_cmd,al
        call    wait
        out     sio_cmd,al
        call    wait
        out     sio_cmd,al
        call    wait
        out     sio_cmd,al      ; ダ ミ ー  コ マ ン ド
        call    wait
        mov     al,40h
        out     sio_cmd,al      ; リ セ ッ ト  コ マ ン ド
        call    wait
        call    wait
        mov     al,1ch
        out     sio_cmd,al      ; 同 期  モ ー ド  セ ッ ト
        call    wait
        call    wait
        mov     al,syn
        out     sio_cmd,al      ; 同 期  キ ャ ラ ク タ  セ ッ ト
        call    wait
        out     sio_cmd,al
        ret
; ダ ミ ー  ウ エ イ ト
wait:
        push    ax
        pop     ax
        ret
```

〈リスト1〉PC9801によるBSC手順(つづき)

```
;*************************************
;*
;*  BSC 受信 ルーチン
;*
;*************************************
recv_entry:
        call    receive         ; 受信
        jb      recv_entry      ; 受信 待ち
        cmp     al,enq          ; 'ENQ' ?
        jnz     recv_entry      ; 'ENQ'待ち
        call    send_ack0       ; 送信 'ACK0'
        mov     snd_ack,1
recv_block:
        call    receive         ; テキスト 受信
        jnb     code_check      ; 受信 正常
        call    send_eot        ; 受信 エラー ('EOT'
        jmp     recv_entry      ; 再受信          送信)
code_check:
        cmp     al,etb          ; 次のブロックあり?
        je      recv_etx        ; yes
        cmp     al,etx          ; テキスト 終了 ?
        je      recv_etx        ; yes
        cmp     al,eot          ; 通信 終了 ?
        je      recv_eot        ; yes
        call    send_nak        ; 'NAK' 送信
        jmp     recv_block      ; ブロック 待ち
recv_etx:
        call    text_save       ;
        call    send_ack        ; 'ACK' 送信
        jmp     recv_block
recv_eot:
        call    send_nak        ; 'NAK' 送信
        ret
;
; データ、コントロール コード 受信&チェック
;
receive:
        mov     al,096h
        out     sio_cmd,al      ; ハント モード セット
        nop
        in      al,sio_stat     ; ダミー リード
        mov     cx,0
recv_hunt:
        in      al,sio_stat     ; ステータス リード
        test    al,18h          ; 受信 エラー ?
        jnz     receive         ; yes
        test    al,40h          ; ハント ok?
        jnz     recv_syn        ; no
        loop    recv_hunt       ;
        stc
recv_err:
        ret
recv_syn:
        call    recv            ; 1キャラクタ・リード
        jb      recv_err        ; 受信 エラー
        cmp     al,syn          ; 'SYN' ?
        je      recv_syn        ; yes
        mov     di,offset recv_buff ; 受信バッファ セット
        cmp     al,stx          ; テキスト ?
        je      recv_text       ; yes
        cmp     al,soh          ; ヘッダ ?
        je      recv_text       ; yes
        jmp     recv_ect        ;
recv_text:
        mov     crc_cnt,0       ; CRC エリア クリア
        mov     recv_len,0      ; 受信 レングス クリア
recv_data:
        call    recv            ; 1文字 受信
        jb      recv_err        ; エラー
        cmp     al,etb          ; 'ETB' ?
        je      recv_tend       ; yes
        cmp     al,etx          ; 'ETX ?
        je      recv_tend       ; yes
        stosb
        inc     recv_len
        call    cal_crc         ; CRC 計算
        jmp     recv_data
recv_tend:                      ; ブロック 終了
        mov     ah,al
        call    cal_crc         ;
        call    recv            ; CRC 受信
        jb      recv_err        ;
        cmp     crc_high,al     ; CRC 比較
        jne     recv_err1       ; CRC エラー
        call    recv            ;
        jb      recv_err        ;
        cmp     crc_low,al      ;
        jne     recv_err1       ; CRC エラー
        mov     al,ah           ; エラー なし
        clc                     ;
        ret
recv_err1:
        stc                     ; 受信 エラー
recv_err2:
        ret
;
```

```
recv_ect:
        stosb
        call    recv
        jb      recv_err2
        cmp     al,padt                 ; 'PADT' ?
        jne     recv_ect                ; no
        mov     al,byte ptr recv_buff   ;
        cmp     al,dle                  ; 'DLE' ?
        jne     recv_ect2               ; no
        mov     al,byte ptr recv_buff+1 ;
recv_ect2:
        clc
        ret
;
;  1文字 受信 ルーチン
;
recv:
        mov     cx,2                    ; タイムアウト レート
recv0:
        push    cx
        mov     cx,0
recv1:
        in      al,sio_stat             ; ステータス リード
        test    al,80h                  ; DSR on ?
        jz      err_term                ; no
        test    al,18h                  ; 受信 エラー ?
        jnz     err_term                ; yes
        test    al,2                    ; 受信 データ あり ?
        jnz     recv2                   ; yes
        loop    recv1
        pop     cx
        loop    recv0
        stc                             ; タイムアウト エラー
        ret
recv2:
        pop     cx
        in      al,sio_data             ; 1 文字 リード
        clc
        ret
err_term:
        pop     cx
        stc                             ; 受信 エラー
        ret
;
;  CRC 計算 ルーチン
;
cal_crc:
        push    ax
        push    dx
        push    cx
        mov     dx,crc_cnt
        mov     cx,0
cal_crc1:
        mov     ah,al
        rcr     ah,1
        and     al,al
        rcr     dx,1
        adc     al,0
        ror     al,1
        jnb     cal_skip
        mov     al,1
        mov     dx,0a001h
cal_skip:
        loop    cal_crc1
        mov     crc_cnt,dx
        pop     cx
        pop     dx
        pop     ax
        ret
;****************************************
;*
;*  BSC 送信 ルーチン
;*
;****************************************
send_entry:
        call    send_enq                ; 'ENQ' 送信
        call    receive                 ; コード 受信
        jb      send_entry
        cmp     al,ack0                 ; 'ACK0' ?
        jne     send_entry              ; no
        mov     ack_flag,1              ; 'ACK' フラグ セット
        mov     send_flag,0             ; 送信 フラグ クリア
send_loop:
        cmp     send_flag,etx           ; 'ETX' 送信 ?
        je      send_exit               ; yes
        call    text_get                ; 送信 データ セット
send_retry:
        call    send_text               ; データ 送信
        call    receive                 ; コード 受信
        jb      send_entry              ; 受信 エラー 再送
        cmp     al,nak                  ; 'NAK' ?
        je      send_retry              ; yes
        mov     ah,ack0                 ; 'ACK0' セット
        cmp     ack_flag,1              ; 送信 'ACK1' ?
        jne     ack_check               ; no
        mov     ah,ack1                 ; 'ACK1' セット
ack_check:
```

〈リスト1〉PC9801によるBSC手順(つづき)

```
        cmp     al,ah               ; 'ACK' 正常 ?
        je      send_loop           ; yes
        xor     ack_flag,1          ; 次の'ACK' セット
        jmp     send_entry          ;
send_exit:
        call    send_eot            ; 送信終了 'EOT' 送信
        ret
; 'ENQ' 送信
send_enq:
        call    send_head           ; 'PADL','SYN'送信
        mov     si,offset enq_id    ; 'ENQ' セット
        call    send_msg            ; 'ENQ' 送信
        ret
; 送信 データ バッファ
send_text:
        call    send_head           ; ヘッダ部 送信
        mov     si,offset send_buff ; 送信 バッファ セット
text1:
        lodsb                       ; データ リード
        call    send                ; 1 文字送信
        cmp     al,padt             ; 'PADT' ?
        jne     text1               ; no
        ret
; 'EOT' 送信
send_eot:
        call    send_head
        mov     si,offset eot_id    ; 'EOT' セット
        call    send_msg            ; 送信
        ret
; ヘッダ部 送信
send_head:
        mov     al,23h              ; RTS on
        out     sio_cmd,al          ; 送信モード セット
        mov     si,offset head_id   ; ヘッダ セット
        call    send_msg            ; 送信 ヘッダ
        ret
; 'ACK' 送信
send_ack:
        cmp     snd_ack,1           ; 'ACK1'送信 ?
        je      send_ack1           ; yes
; 'ACK0' 送信
send_ack0:
        call    send_head
        mov     si,offset ack0_id   ; 'ACK0' セット
        call    send_msg            ; 送信
        ret
; 'ACK1' 送信
send_ack1:
        call    send_head
        mov     si,offset ack1_id   ; 'ACK1' セット
        call    send_msg            ; 送信
        ret
; 'NAK' 送信
send_nak:
        call    send_head
        mov     al,nak              ; 'NAK' セット
        call    send                ; 送信
        mov     al,padt             ; 'PADT' セット
        call    send                ;
        ret
; 'WACK' 送信
send_wack:
        call    send_head
        mov     si,offset wack_id   ; 'WACK' セット
        call    send_msg            ; 送信
        ret
;
send_msg:
        lodsb                       ; 送信 レングス
        mov     cl,al
send_msg1:
        lodsb                       ; データ リード
        call    send                ; 1 文字送信
        jb      send_err            ; 送信 エラー
        dec     cl
        jnz     send_msg1
send_err:
        ret
;
; 1 文字送信
;
send:
        mov     ah,al
        push    cx
        mov     cx,0                ; タイムアウト セット
send1:
        in      al,sio_stat         ; ステータス リード
        test    al,80h              ; ターミナル エラー ?
        jz      term_err            ; yes
        test    al,1                ; 送信 ok ?
        jnz     send2               ; yes
        loop    send1               ; 送信 待ち
term_err:
        pop     cx
        stc                         ; 送信 エラー
        ret
```

〈リスト1〉PC9801によるBSC手順(つづき)

```
send2:
        pop     cx
        mov     al,ah                   ; データ  セット
        out     sio_data,al             ;  送信
        clc                             ;
        ret
; コントロール  文字  テーブル
head_id         db      3
                db      padl
                db      syn,syn
enq_id          db      2
                db      enq
                db      padt
eot_id          db      2
                db      eot
                db      padt
ack0_id         db      3
                db      dle
                db      ack0
                db      padt
ack1_id         db      3
                db      dle
                db      ack1
                db      padt
wack_id         db      3
                db      dle
                db      wack
                db      padt
;
text_save:
        push    es
        les     di,text_addr            ; ユーザ  バッファ
        mov     si,offset recv_buff     ; 受信  バッファ
        mov     cx,recv_len             ; 受信  レングス
        add     text_len,cx             ;
        rep     movsb                   ; ユーザ  バッファに転送
        mov     text_off,di             ; 次のバッファ
        pop     es
        ret
;
text_get:
        call    len_get                 ; 送信  レングス  (CX)
        mov     di,offset send_buff     ; 送信  バッファ
        mov     al,stx
        stosb                           ; 'STX'  付加
        jcxz    text_end                ; データ  なし
        mov     crc_cnt,0               ; CRC エリア  クリア
        push    es
        les     si,text_addr            ; ユーザ  バッファ  セット
text_in:
        mov     al,es:[si]
        mov     [di],al                 ; 送信  バッファに転送
        call    cal_crc                 ; CRC  計算
        inc     si
        inc     di
        loop    text_in
        pop     es
        cmp     text_len,0              ; ユーザ  テキスト  終了  ?
        je      text_end                ; yes
        mov     text_off,si
        mov     al,etb                  ; 'ETB'  セット
text_code:
        mov     send_flag,al            ; 送信  コード
        stosb                           ;
        call    cal_crc                 ; CRC  計算
        mov     al,crc_high
        stosb                           ; CRC1  セット
        mov     al,crc_low
        stosb                           ; CRC2  セット
        mov     al,padt
        stosb                           ; 'PADT'  セット
        ret
text_end:
        mov     al,etx                  ; 'ETX'  セット
        jmp     text_code
;
; CX:送信レングス
len_get:
        mov     cx,256
        cmp     text_len,cx             ; 256  以上  ?
        jnb     len_end                 ; yes
        mov     cx,text_len
len_end:
        sub     text_len,cx
        ret
code    ends
        end     start
```

§2-4

RS-232Cチェッカの製作

鶴野和孝

最近はほとんどのパソコンにシリアル・インターフェースとしてRS-232Cが標準装備されるようになりました(図1,表1参照). ところが, これに周辺装置を接続する場合, カタログどおりの組み合わせ(いわゆる純正)ならまず問題はありませんが, そうでない場合にはうまく接続できないことがあります.

カタログや仕様書を理解しているつもりでも, ちょっとした事柄を見落としていたり, ときには仕様書そのものが不完全だったりします. とくにパソコンのマニュアルは厚いので, 見過ごしが多くなります.

トラブルがあった場合, ロジック・アナライザなどの測定器が, 気軽に使用できる環境ならよいのですが, 多くの人はそうでないと思います. そこで簡単なチェッカまたは治具があれば便利と考え, 今回チェッカを製作しました.

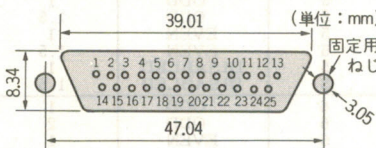
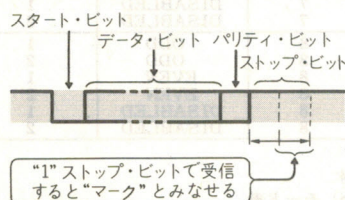
● トラブルの原因

機器間が相互にマッチングしなければならない条件として,

(1) ボーレート
(2) 信号論理
(3) 信号レベル
(4) ストップ・ビット数
(5) パリティ
(6) ハンドシェイクのタイミング
(7) ソフト的なデリミタ

などが考えられますが, これらの違いがどのような現象として現れるか考えてみます.

(1) ボーレートの違いは, フレーミング・エラー, パリティ・エラーおよび送受データの違いとして現れる.
(2) データ・ラインの論理が逆だと, 送信していないときにブレーク・キャラクタとして受信される. ハンドシェイク線(DSR,DTR,RTS,CTSなど)の論理が逆の場合は, 送るべきときに送らず, 送っていけないときに送るということになってしまう.
(3) 信号のレベルが合わなければ, まったく動かないとか, 悪ければICを壊してしまうこともある.
(4) ストップ・ビット数がわからないときは"1"で受信するとうまくいく(図2参照). これは, もし相手が"2"ストップ・ビットで送信しても, 二つめのストップ・ビットをマークと見なすから.
(5) パリティのODD/EVENの違いは, 当然パリティ・エラーで, パリティ・ビットの有無の違いはデータ・ビット長に影響し, 送受データの違いとして現れたり, パリティ・エラーを発生したりする.

〈図1〉分界線に使用するコネクタ

〈図2〉シリアル・データのフォーマット

〈表1〉信号名称表

ピン番号	名称	働き
1	FG	フレーム・グラウンド. 筐体のグラウンド
2	TxD	送信データ. 端末からモデムへデータを送信する
3	RxD	受信データ. 端末はモデムからのデータを受信する
4	RTS	送信要求. 端末がデータを送出したい時ONにする
5	CTS	送信許可. 端末がデータを送出してよいかどうかをモデムが知らせる
6	DSR	データ・セット・レディ. モデムが送受可能であることを端末に知らせる
7	SG	信号用グラウンド
8	CD	キャリア検出. モデムが回線からキャリアを受信すると, ONを端末につたえる
20	DTR	端末レディ. 端末が送受可能であることをモデムに知らせる

(注) このほかのピンも名称や働きが決められているが, ここには一般的なものだけ書き出した

(6) ハンドシェイクのタイミングが合わないと，データが失われる場合がある．

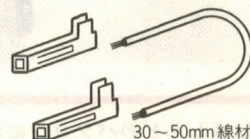
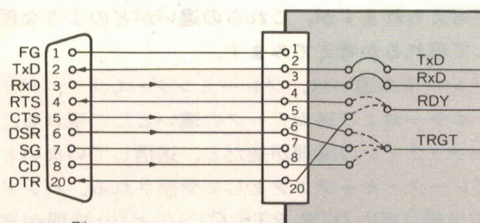
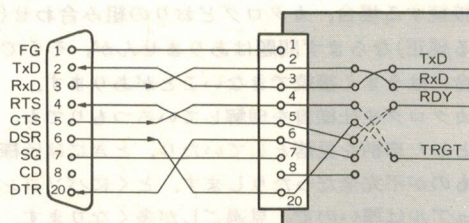
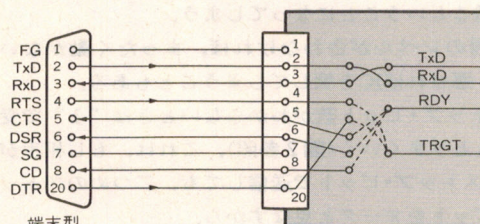
(7) デリミタの有無，および食い違いは，それぞれの機器のソフトの働きに依存する．

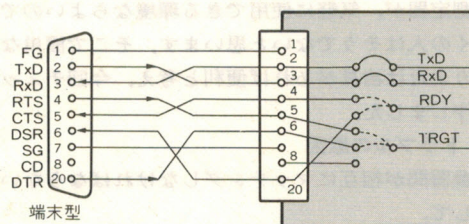
以上のことから，オシロスコープやテスタではわかりにくい事象を拾ってみると，パリティのチェック，ハンドシェイクのタイミング，ソフト的なデリミタの有無の3点になります．これらを重点にチェックする方法を考えたいと思います．

チェッカの操作方法

まず相手の信号の入出力の確認のうえ，チェック・ピンをショート・ワイヤで接続します(図3)．相手がモデム型の入出力になっていれば，TxDとRxDはそのままストレートに，RDYはRTSまたはDTRに，TRGTはCTS，DSRまたはCDに接続します．信号グラウンドSGは，パターン上で接続されています．

なお上記は，接続ケーブルが1対1としての話です．ケーブル内でクロスなどの処理がされている場合は，

〈図3〉ショート・ワイヤ

〈図4〉モデム型と端末型の接続

(a) ストレート・ケーブル/モデム型の場合

(b) ストレート・ケーブル/端末型の場合

(c) クロス・ケーブル/モデム型の場合

(d) クロス・ケーブル/端末型の場合

〈表2〉ディップ・スイッチの設定

▶DIP SW_1

4	3	2	1	ボーレート(ボー)
ON	ON	OFF	ON	50
ON	ON	OFF	OFF	75
ON	OFF	ON	ON	134.5
ON	OFF	ON	OFF	200
ON	OFF	OFF	ON	600
ON	OFF	OFF	OFF	2400
OFF	ON	ON	ON	9600
OFF	ON	ON	OFF	4800
OFF	ON	OFF	ON	1800
OFF	ON	OFF	OFF	1200
OFF	OFF	ON	ON	2400
OFF	OFF	ON	OFF	300
OFF	OFF	OFF	ON	150
OFF	OFF	OFF	OFF	110

(a) ボーレート表

▶DIP SW_2

DIP SW					データ・ビット長	パリティ	ストップ・ビット長
5	4	3	2	1			
L	L	L	L	L	5	ODD	1
L	L	L	L	H	5	ODD	1.5
L	L	L	H	L	5	EVEN	1
L	L	L	H	H	5	EVEN	1.5
L	L	H	×	L	5	DISABLED	1
L	L	H	×	H	5	DISABLED	1.5
L	H	L	L	L	6	ODD	1
L	H	L	L	H	6	ODD	2
L	H	L	H	L	6	EVEN	1
L	H	L	H	H	6	EVEN	2
L	H	H	×	L	6	DISABLED	1
L	H	H	×	H	6	DISABLED	2
H	L	L	L	L	7	ODD	1
H	L	L	L	H	7	ODD	2
H	L	L	H	L	7	EVEN	1
H	L	L	H	H	7	EVEN	2
H	L	H	×	L	7	DISABLED	1
H	L	H	×	H	7	DISABLED	2
H	H	L	L	L	8	ODD	1
H	H	L	L	H	8	ODD	2
H	H	L	H	L	8	EVEN	1
H	H	L	H	H	8	EVEN	2
H	H	H	×	L	8	DISABLED	1
H	H	H	×	H	8	DISABLED	2

(注1) ×＝Don't Care
(注2) "L"＝ON，"H"＝OFFを示す

(b) モード表

この限りではありません(図4参照).

表2で示すボーレート表，モード表によりDIP SW_1，DIP SW_2をセットします．これらのディップ・スイッチを変更した場合は，必ずMODE(SW)を押して，モードの再設定を行う必要があります．

● 受信操作

DISPLAY FIFO/RxD(SW)をRxD側に，Rx FREE/STOP(SW)をSTOP側に倒します．

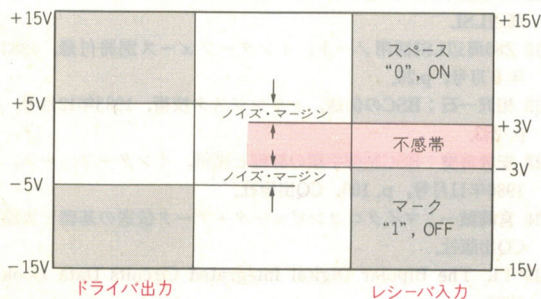
次にターゲット機器から何かデータを送信します．例えば"ABCDEF",CR,LFを送信したとすると，このチェッカのRx STEP(SW)を押すたびにRB_0～RB_7(LED)に"B","D","F",LFと，ひとつおきに表示されます．これは8251AなどのUSARTの特徴のようで，今回使用している機器のシリアル・ポートは，4機種とも外部からビジィ状態を送り込んでも直ぐには送信停止にはならないで，必ず1バイト余計に送ってきます．

ここで，DISPLAY FIFO/RxD(SW)をFIFO側にすると，RB_0～RB_7(LED)には16バイトのFIFOの内容が，Rx STEP(SW)を押すごとに"A","B","C",…と，受信時のこのチェッカのビジィ状態とともに表示されます．このチェッカのビジィは，READY表示部のRECEIVE(LED)の消灯により表示されます．また，DISPLAY FIFO/RxD(SW)がRxD側にあるときは，ビジィ/レディにかかわらず消灯します．下記に述べるSTATUS(LED)も同様です．

また，STATUS IN(PIN)にPE(PIN)，OE(PIN)，FE(PIN)を接続しておいたならば，同様にデータ受信時のエラー状態をもFIFOに取り込まれ，DISPLAY FIFO/RxD(SW)をFIFO側に倒すと，STATUS(LED)により表示されます．

● 送信操作

ターゲットの機器にデータを送るには，READY表示部のSEND(LED)とTARGET(LED)がともに点灯しているときに，TB_0～TB_7(SW)により送るべきデータを設定し，Tx STEP(SW)を押すことによりデータが送信されます．

〈図5〉RS-232Cのレベル

回路構成

▶LSI

まず，シリアル通信の中心であるデータのアセンブル/ディスアセンブルはLSIに任せることにします．インターシル社のIM6402を使用するとパリティ・チェックやスタート/ストップ・ビットの検出など，シリアル通信に不可欠な機能が全部LSI任せとなって手間が省けます．

IM6402はパリティのイネーブル/ディセーブルおよびODD/EVEN，ストップ・ビット数，データ・ビット長の設定が外部よりできるようになっていますので，ここにディップ・スイッチを接続して，任意に変更できるようにします．

▶ボーレート

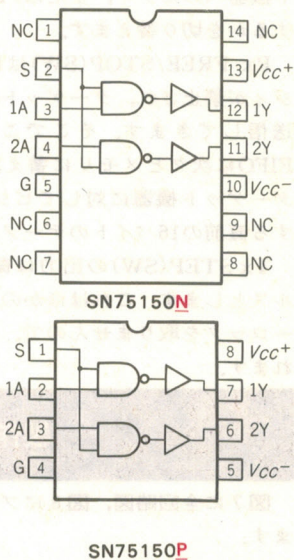
ターゲット機器によってボーレートが違いますので，これも変更できるようにしておきます．やはりインターシル社のIM4712/02を使うと，14種類のボーレート・クロックが取り出せます．IM4712を使用すると回路図中のR_1, C_6, C_7は実装しなくてもかまいません．

▶ドライバ/レシーバ

ライン・ドライバにはSN75150を使用します．これはRS-232C用で，±12Vの電源を供給すると規格に合う出力特性が得られます(図5)．このICにはSN75150PとSN75150Nの2種があり，PとNではピン数が違います(図6)．しかし中身は同じで，ピン配列も似ていますので，緊急の場合は差し替えることもできます．

▶チェック・ピン

RS-232Cの信号の使い方には，モデム型と端末型があり，それぞれを接続するのにストレート型，クロ

〈図6〉[15] レベル変換用IC

ス型，変形型などいろいろな接続方法が使われています．これら全部に対応するために，スイッチを使って切り替えていたら大変なことになります．

そこで一番簡単な方法として，Dサブ・コネクタとドライバ/レシーバとの接続はチェック・ピンを立てて，ショート・ワイヤを使用することにしました．ほとんどの場合はTxD, RxD, DTR(またはRTS)，DSR(またはCTS)の4本ですみます．また，FIFOのビット数が少ないので，オーバラン・エラー(OE)，パリティ・エラー(PE)，フレーミング・エラー(FE)もチェック・ピンを立ててショート・ワイヤでSTATUS IN(PIN)に接続します．

▶表示

受信データ(RB_0～RB_7)を点LEDを使ってバイナリ表示します．また，1バイトでもデータが受信されると，必ずFIFOに記憶され，FIFO REMAIN(LED)が点灯します．

RB_0～RB_7(LED)は，DISPLAY FIFO/RxD(SW)によりFIFOの出力またはUARTの出力を表示します．

STATUS(LED)はショート・ワイヤで，STATUS IN(PIN)に入力されたPE, OE, FEの状態を表示します(エラーで点灯)．

READY表示としてRECEIVE(LED)，SEND(LED)を用意し，このチェッカの状態を表示し，TERGET(LED)でターゲット機器の状態を表します．

▶操作スイッチ

RxSTEP(SW)，TxSTEP(SW)，Rx FREE/STOP(SW)はチャタリングを考慮し，RSフリップフロップを使います．

DISPLAY FIFO/RxD(SW)は上に倒すとFIFOの内容を，下に倒すとUARTの出力をRB_0～RB_7(LED)に表示し，Rx STEP(SW)の出力をターゲット機器へのビジィ，またはFIFOへのリード・パルスとするかを切り替えます．

Rx FREE/STOP(SW)はFREE側(上)に倒すとビジィが禁止され，ターゲット機器はデータをどんどん送信してきます．そこでこのチェッカは，それをFIFOに次々とメモリに蓄え，STOP側(下)に倒すとターゲット機器に対してビジィとなって，"STOP"する直前の16バイトのデータをFIFOに残します．

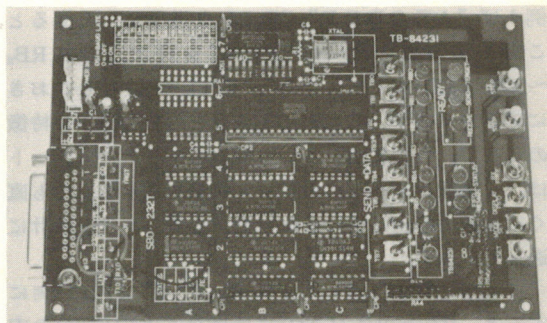
Tx STEP(SW)の出力は微分し，UARTの送信パルスとします．これはほかのどのスイッチともインターロックを取りませんので，受信と独立に動作させられます．

製作

図7に全回路図，図8にプリント・パターンを示します．

製作上難しいところはないと思います．DIP $SW_{1,2}$を実装するとき，回路No.の順序に気をつけてください．逆に取り付けると，ボーレート表やモード表との対応ができなくなります．

また，プッシュ・スイッチのN. O.とN. C.を逆さにするとメチャメチャな動作になります．

IM4712がなくてIM4702を使う人はR_1=10MΩ, C_6=C_7=56pFを忘れずに取り付けてください．

〈写真1〉チェッカの外観

◆参考・引用*文献◆

(1)*神崎康宏；特集＊マイコン設計技術の完全マスタ，トランジスタ技術，1985年5月号．

(2) 森野ひとみ；マイコンとデータ伝送，トランジスタ技術，1983年12月号．

(3) 特集＊マイコン周辺LSI完璧マスタ，トランジスタ技術，1985年3月号．

(4) 特集＊実験で学ぶディジタルIC回路，トランジスタ技術，1984年2月号．

(5) 石井裕次；シリアル・インターフェースの設計法，トランジスタ技術，1985年11月号．

(6) 相良富美；周辺制御回路(シリアル・インターフェース)の設計と検討，トランジスタ技術，1982年3月号．

(7)*PERIPHERAL DESIGN HANDBOOK, INTEL.

(8)*Z80 SIOテクニカル・マニュアル，シャープ．

(9)*日本電気，PC9801ユーザーズマニュアル(VF/VM/VX)．

(10)*日立製作所，日立8ビット・16ビットマイクロコンピュータ周辺LSI.

(11) Z80周辺LSI活用ノート，インターフェース別冊付録，1983年5月号，p.25.

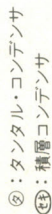
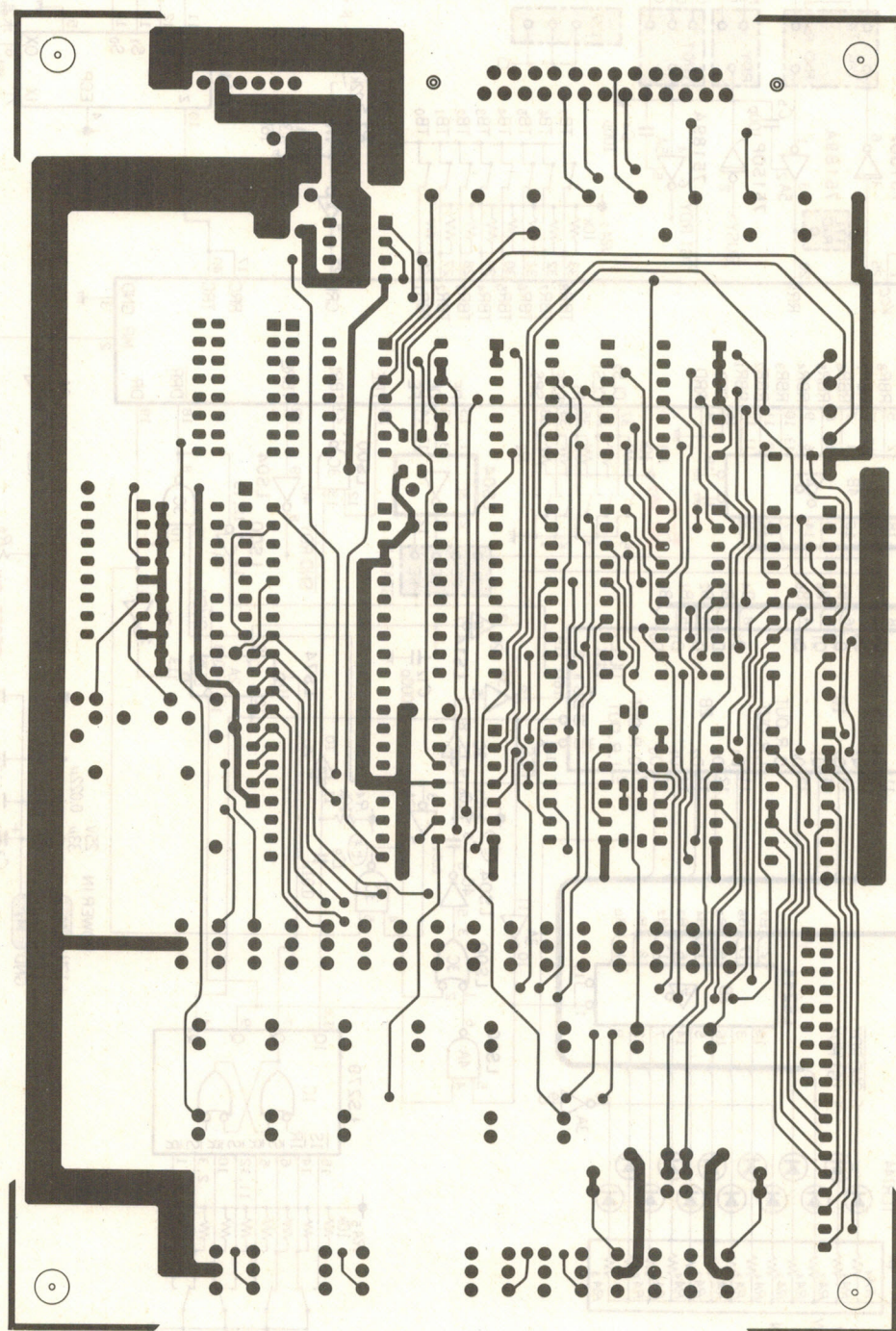
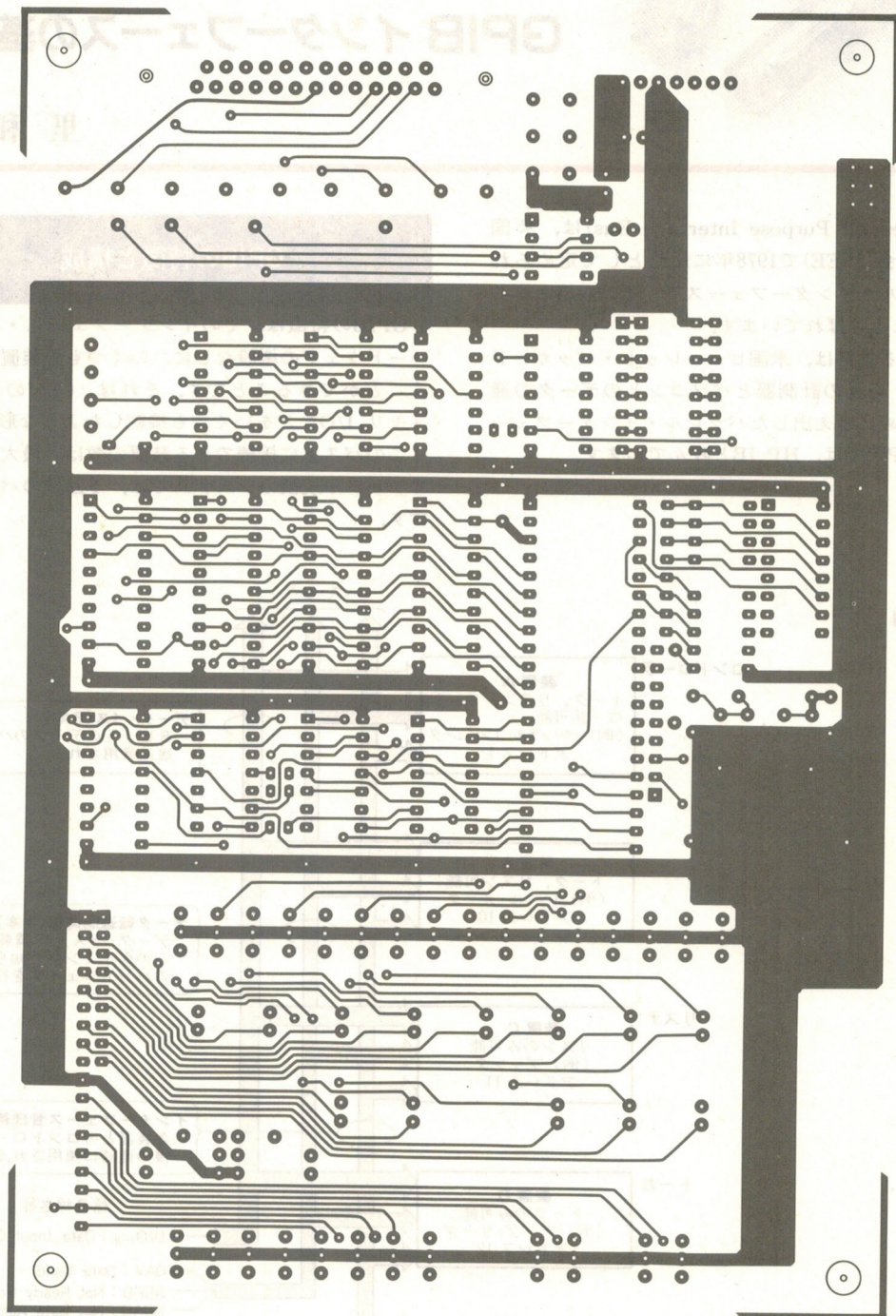
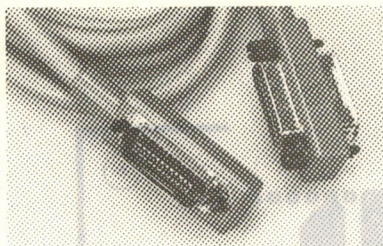
(12) 相沢一石；BSCの伝送，トランジスタ技術，1983年12月号，p. 283.

(13) 安藤善廣；BSC制御手順の詳細と実例，インターフェース，1980年11月号，p. 100，CQ出版社．

(14) 宮崎誠一；マイクロコンピュータ・データ伝送の基礎と実際，CQ出版社．

(15)*TI, The Bipolar Digital Integrated Circuits Data Book, 1986.

〈図7〉RS-232Cのチェッカの全回路

㋟：タンタル・コンデンサ
㋝：積層コンデンサ

〈図8(a)〉プリント板のはんだ面(原寸)

〈図8(b)〉 プリント板の部品面(原寸)

§3-1

GPIBインターフェースの基礎

里　和政

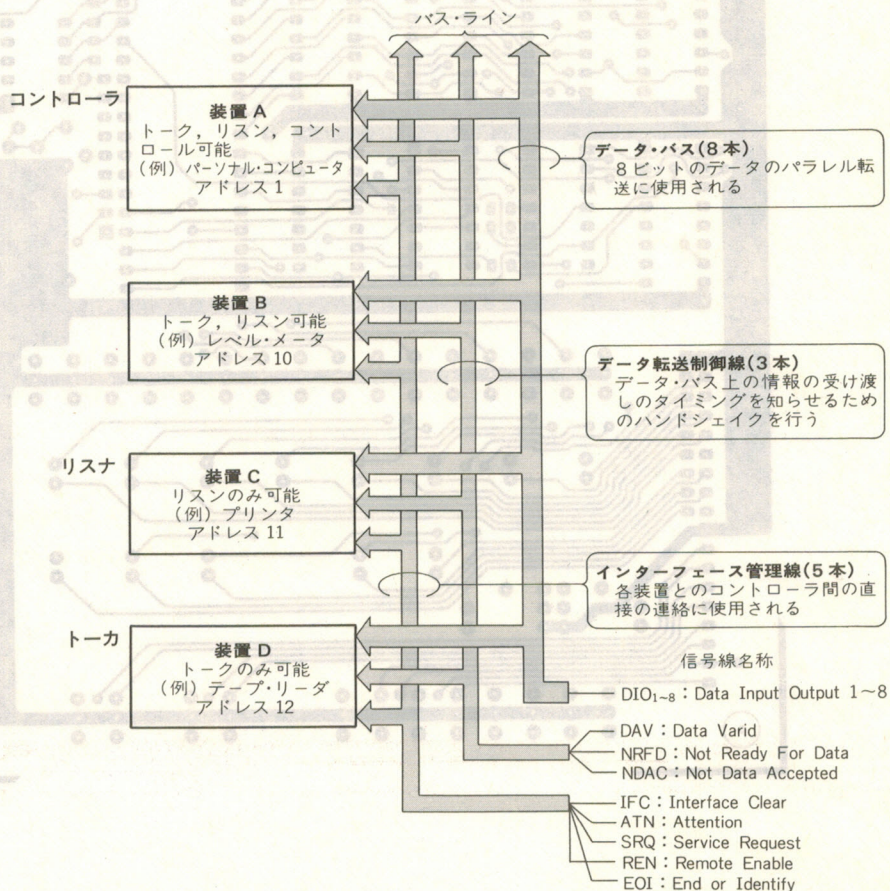
GPIB(General Purpose Interface Bus)は，米国電気電子学会(IEEE)で1978年に規格として定められたディジタル・インターフェースで，その規格名からIEEE-488とも呼ばれています．

基本となる規格は，米国ヒューレット・パッカード社(HP)が，自社の計測器とパソコンとのデータの通信を行うために考え出したパラレル・インターフェースです．HP社では，HP-IBと呼んでいます．

GPIBの特徴と構成

GPIBの特徴は，そのインターフェース・バス上にハードウェアの増設なしに，いくつもの装置を追加することができることです．それは，CPUのバス上にメモリ,I/Oなどをいくつも接続したような形です．

このバス上に接続できる装置の数は，最大15台までですが，中規模のシステムでは，１回線のバスで十分です．

〈図１〉[3]
GPIBの構成例

〈表1〉[(2),(26)] GPIBの信号線とその機能

	信号線	機能	
データ・バス	DIO1 (Data Input/Output 1) DIO2 (Data Input/Output 2) DIO3 (Data Input/Output 3) DIO4 (Data Input/Output 4) DIO5 (Data Input/Output 5) DIO6 (Data Input/Output 6) DIO7 (Data Input/Output 7) DIO8 (Data Input/Output 8)	データの伝達 (データ例) コマンド アドレス 測定データ ステータス	
伝送制御線	DAV (Data Valid) NRFD (Not Ready For Data) NDAC (Not Data Accepted)	データ有効 受信準備完了 (NRFD=0の時) 受信完了 (NDAC=0の時)	ハンドシェイクを行う
管理線	ATN (Attention) IFC (Interface Clear) SRQ (Service Request) REN (Remote Enable) EOI (End or Identify)	データの区別をする(1:インターフェース・メッセージ 0:デバイス・メッセージ) インターフェースを初期化する サービス要求 リモート/ローカル切り替え データの最終バイトを示す (ATN=0の時) パラレル・ポールの実行を示す (ATN=1の時)	

(注 バスはすべて負論理. 0="H", 1="L"レベル)

それらに加えてGPIBは，制御コマンドも決められています．

図1にGPIBの構成を示します．ここで，トーカは送信のみを行う機器であり，バス上に複数のトーカを接続することが可能ですが，同時に複数の動作はできません．

リスナは，受信のみを行う機器であり，これもトーカ同様に複数のリスナをバスに接続することができます．ただし，トーカとは異なり，同時に複数のリスナを動作させることが可能です．

コントローラは，バスの制御を行い，各装置(トーカ，リスナ)からの要求またはデータの送受信を行います．

また，これらの機能をすべてもった装置などもあります．

◈ GPIBのバス構成

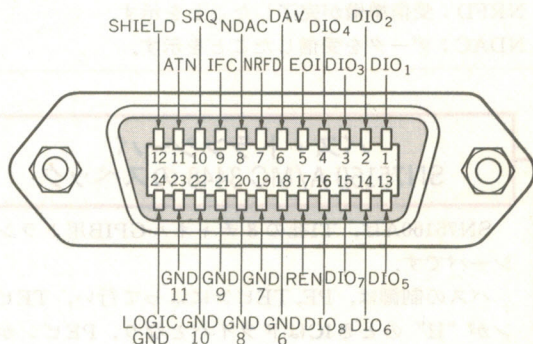
GPIBバスは，8本のデータ・バス，3本の伝送制御線と5本の管理線の16本から構成されています．表1に各信号線を示します．

データ・バスは，双方向性でパラレルにデータを伝送します．伝送制御線(ハンドシェイク・バス)は，データ・バス上のデータの伝送タイミングおよびデータの方向を制御し，管理線はコントローラが制御する信号で，インターフェースの初期化，データの区別，割り込み，メッセージの管理などを行います．

◈ GPIBの電気的特性

GPIBでは，コネクタの形状，信号の制御方法，電気的特性などが規格化されています．図2にコネクタを示します．

電気的特性は，接続できる装置の数，ケーブルの長さ，転送速度，装置のバス・ドライブ能力などがあります．

〈図2〉[(2),(26)] GPIBコネクタ

[例……IEEE規格コネクタ IEC規格とは異なる]

(1) 装置の接続できる数は，前にも述べたように最大15台です．
(2) 接続するケーブルの長さは，各装置間で4m以内で，転送データの信頼性などにより，装置の数が11台以上の場合はその総合計が20m以内で，装置が10台以下の場合では，ケーブルの長さは装置の数を2倍した値以下です．
(3) 転送速度は，1Mバイト/秒以内ですが，装置の接続数によって負荷が異なるため，転送速度は遅くなります．
(4) GPIBのバス・ドライバには，オープン・コレクタまたは3ステート・ドライバが使用(NRFD, NDAC, SRQはオープン・コレクタのみ)され，レシーバの入力電圧は"L"レベルで0.8V以下，"H"レベルで2.0V以上となっています．

ドライバの出力電圧は，"L"レベルで0.5V以下(電

流シンク＋48mA)，“H” レベルで2.4V以上(−5.2 mA)となっています．

これらの規格にあった専用のドライバ/レシーバIC (SN75160Aなど)があり，ノイズ・マージンなどもとれ，これらを用いるのがよいでしょう．

GPIBのケーブルを長く延ばす場合は，コンバータでRS-232Cや光ファイバなどに変換して転送し，再度GPIBにもどす方法があります．

ハンドシェイクの方法

GPIBでは，複数の装置を制御するため少し複雑な転送制御を行います．その手順をハンドシェイクと呼び，DAV, NRFD, NDACの３本の制御信号線によってコントロールします．

以下，ハンドシェイクの手順を示します．

DAV：送信されたデータが有効であることを示す．
NRFD：受信準備が完了したことを示す．
NDAC：データを受信したことを示す．

DAVを “H” にし，それによって受信側は，NRFD, NDACを “L” にします．

送信側は，データをデータ・バス上に出力します．受信側すべての装置を受信完了でNRFDが “H” となります．

送信側は，NRFDが “H” となると，DAVを “L” にしてデータを有効にします．受信側は，DAVが “L” になったのちNRFDを “L” にして，データを受信します．受信が完了した時点で，NDACを “H” にします．

送信側は，NDACが “H” になったのちにDAVを “H” にして，次のデータの送信を行います．DAVが “H” になったのち受信側は，NDACを “L” にします．これらによって１バイトの転送が完了します．

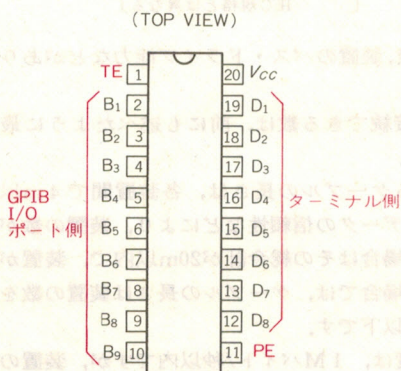
この手順を繰り返すことによって，連続転送を行います．図3に転送タイミングを示します．

GPIBのコマンド

GPIBは，コントローラからトーカ，リスナにコマ

バス・トランシーバ SN75160 A/MC 3448 のスペック

SN75160Aは，TI社の８チャネルGPIB用トランシーバです．

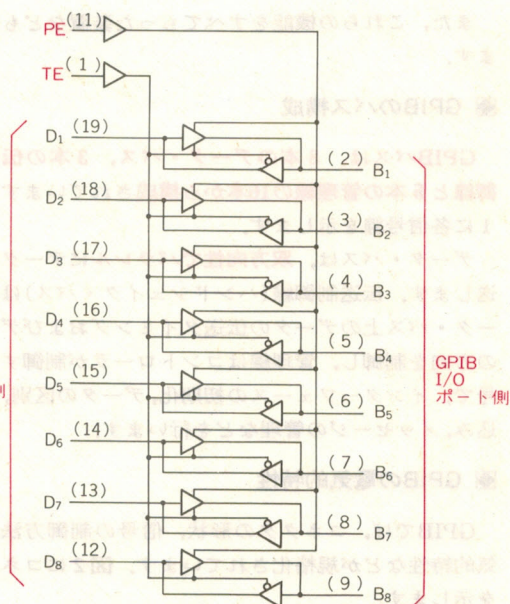
バスの制御は，PE, TEピンによって行い，TEピンが “H” のときICはドライバとなり，PEピンが “H” で出力できます．TEピンが “L” のときは，レシーバとなります．図Aにピン配置，図Bに内部構成を，表Aに電気的特性を示します．

ほかに，同様のICとしてSN75162Aがあります．このICは，SN75160Aとほぼ同一仕様ですが，ピンが

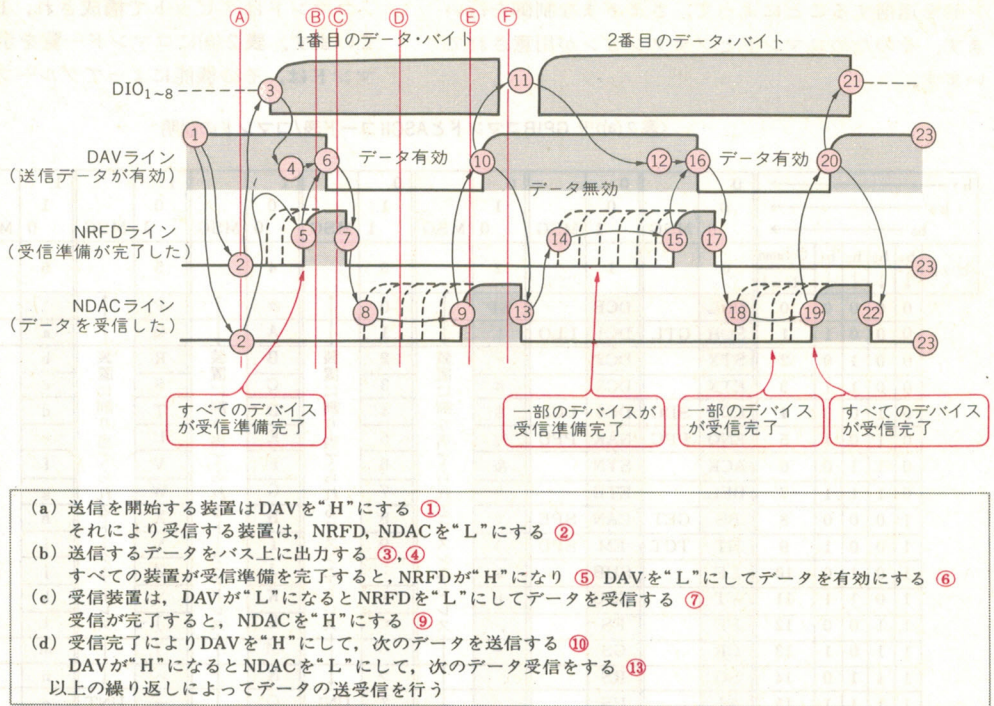
〈図A〉(7) SN75160Aのピンの配置

(TOP VIEW)

左側	ピン	ピン	右側
TE	1	20	V_{CC}
B_1	2	19	D_1
B_2	3	18	D_2
B_3	4	17	D_3
B_4	5	16	D_4
B_5	6	15	D_5
B_6	7	14	D_6
B_7	8	13	D_7
B_8	9	12	D_8
B_9	10	11	PE

GPIB I/O ポート側（B_1～B_8）　ターミナル側（D_1～D_8）

〈図B〉(7) SN75160Aの内部構造

〈図3〉[2],[26] GPIBの3線ハンドシェイクのタイムチャート

(a) 送信を開始する装置はDAVを"H"にする ①
それにより受信する装置は，NRFD, NDACを"L"にする ②
(b) 送信するデータをバス上に出力する ③, ④
すべての装置が受信準備を完了すると，NRFDが"H"になり ⑤ DAVを"L"にしてデータを有効にする ⑥
(c) 受信装置は，DAVが"L"になるとNRFDを"L"にしてデータを受信する ⑦
受信が完了すると，NDACを"H"にする ⑨
(d) 受信完了によりDAVを"H"にして，次のデータを送信する ⑩
DAVが"H"になるとNDACを"L"にして，次のデータ受信をする ⑬
以上の繰り返しによってデータの送受信を行う

GPIBバスと同じ形式となっています(図C参照).

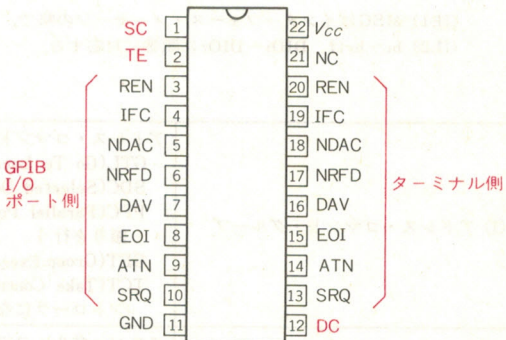
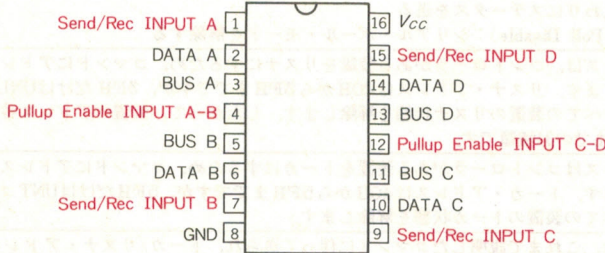
MC3448は，モトローラ社の4チャネルGPIB用トランシーバです.

SN75160Aと同じように Send/Rec.,Enableピンによって制御します. Send/Rec.ピンは，"H"のときドライバに，"L"のときレシーバになります. Enableピンは，ドライバ動作時に"L"で出力されます.

図Dにピン配置を示します.

〈表A〉 SN75160Aの電気的特性

電源電圧	＋5V
消費電力	66mW
ドライバ部	オープン・コレクタ
ドライブ電流	48mA
レシーバ・ヒステリシス	650mV

〈図C〉[7] SN75162Aのピンの配置

〈図D〉[6] MC3448Aのピンの配置

電源電圧	+5V
ドライバ・ドライブ電流	48mA
レシーバ・ヒステリシス	400mV

ンドを送信することによって，さまざまな制御を行います．そのためコマンドとファクションが用意されています．

コマンドは7ビットで構成され，128種類から成っています．表2(a)にコマンド一覧を示します．またコマンドは，その機能によってグループごとに区別され

〈表2(a)〉[1] GPIBコマンドとASCIIコード表/コマンドの説明

b7 → b6 → b5 →					0 0 0	MSG	0 0 1	MSG	0 1 0	MSG	0 1 1	MSG	1 0 0	MSG	1 0 1	MSG	1 1 0	MSG	1 1 1	MSG
ビット b4↓	b3↓	b2↓	b1↓	Column / Row↓	0		1		2		3		4		5		6		7	
0	0	0	0	0	NUL		DCE		SP	↑ 装置に割り当てられるリスン・アドレス ↓	0	↑ 装置に割り当てられるリスン・アドレス ↓	@	↑ 装置に割り当てられるトーク・アドレス ↓	P	↑ 装置に割り当てられるトーク・アドレス ↓	'	↑ PCGによって決定されて意味をもつ ↓	p	↑ PCGによって決定されて意味をもつ ↓
0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
0	1	0	1	5	ENQ	PPC	NAK	PPU	%		5		E		U		e		u	
0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	7	BEL		ETB		'		7		G		W		g		w	
1	0	0	0	8	BS	GET	CAN	SPE	(		8		H		X		h		x	
1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1	0	1	0	10	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	11	VT		ESC		+		;		K		＼		k		{	
1	1	0	0	12	FF		FS		,		<		L		〕		l		¦	
1	1	0	1	13	CR		GS		—		=		M		〕		m		}	
1	1	1	0	14	SO		RS		.		>		N		^		n		~	
1	1	1	1	15	SI		US		／		?	UNL	O		—	UNT	o		DEL	

アドレス・コマンド・グループ　ユニバーサル・コマンド・グループ　リスナ・アドレス・グループ　トーカ・アドレス・グループ

一次コマンド・グループ（PCG）

二次コマンド・グループ（SCG）

（注1）MSGはインターフェース・メッセージの略で，これらはATN="1"で送出される．
（注2）b_1～b_7は，DIO_1～DIO_7に順番に対応する．

① アドレス・コマンド・グループ	アドレス・コマンドは，リスナまたはトーカに指定されている装置が機能するコマンドです． GTL(Go To Local)：ローカル状態になる SDC(Selected Device Clear)：装置の初期化を行う PPC(Parallel Poll Configure)：二次コマンドと併用し，パラレル・ポールのライン割り振りを行う GET(Group Execute Trigger)：トリガ(測定開始)を受けることになる TCT(Take Control)：トーカに指定され，このコマンドを受信すると，その装置が以後コントローラになる
② ユニバーサル・コマンド・グループ	ユニバーサル・コマンドは，バスに接続されているすべての装置に対して送られるコマンドです． LLO(Local Lockout)：ローカル・ロック状態になり，装置側からリモート/ローカル状態を変えられなくなる DCL(Device Clear)：すべての装置が初期化を行う PPU(Parallel Poll Unconfigure)：すべての装置のパラレル・ポールの設定をクリアする SPE(Serial Poll Enable)：GPIBをシリアル・ポール・モードにする．このモード中は，データの代わりにステータスを送る SPD(Serial Poll Disable)：シリアル・ポール・モードを解除する
③ リスナ・アドレス・グループ	リスナ・アドレスは，コントローラがある装置をリスナにするため，コマンドにアドレス番号を付けて送ります．リスナ・アドレスは20Hから3FHまでですが，3FHだけはUNLコマンドとして，すべての装置のリスナ状態を解除します．したがって，装置のアドレス番号は00Hから1EHまでの31種類です．
④ トーカ・アドレス・グループ	トーカ・アドレスはコントローラがある装置をトーカにするため，コマンドにアドレス番号を付けて送ります．トーカ・アドレスは40Hから5FHまでですが，5FHだけはUNTコマンドとして，すべての装置のトーカ状態を解除します．
⑤ 二次コマンド・グループ	二次コマンドは，これまで説明したコマンドに伴って送られ，トーカ/リスナ・アドレスを拡張したり，パラレル・ポールのライン割り振りに使用します．

ています(コマンド説明参照)．動作の対象となる装置は，コマンドによって１台または複数台となります．表２(b)にコマンドの送信シーケンスを示します．

コマンドを送る場合は，ATN信号を“H”にして装置にコマンドであることを認識させます．データの場合は，ATN信号は，“L”です．

ファンクションは10種類あり，表３に示します．10種類のどのファンクションを使用するかは，設計者が必要に応じて決めます．

コマンドは，コントローラがGPIB装置を制御するために用いられ，ファンクションは，装置の拡張機能であり，高度な動作を行う場合に使用します．

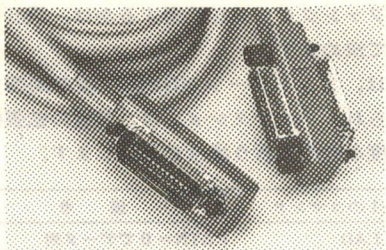
〈表２(b)〉[3]
コマンド送信のシーケンス

項　目	ATN＝１のコマンド	備　考
データ転送	UNL, TAD, LAD	ATN＝０でデータ列
リモート	UNL, LAD	事前にREN＝１にする
ローカル	UNL, LAD, GTL	事前にREN＝１にする
ローカル・ロックアウト	UNL, LLO	事前にREN＝１にする
トリガ	UNL, LAD, GET	
デバイス・クリア	UNL, DCL	全装置が対象
デバイス・クリア	UNL, LAD, SDC	指定装置が対象
コントローラ委譲	TAD, TCT	
シリアル・ポール実行	UNL, SPE, CLA, TAD	ATN＝０でステータス・バイト
シリアル・ポール終了	UNL, SPD	
パラレル・ポールの設定	UNL, LAD, PPC, PPE, UNL	
パラレル・ポールの解除	PPU	全装置が対象
パラレル・ポールの解除	UNL, LAD, PPC, PPE, UNL	指定装置が対象
パラレル・ポールの実行	パラレル・ポールの応答データ	ATN＝１に加えてEOI＝１

- GTL (Go To Local)
- LLO (Local Lockout)
- GET (Group Execute Trigger)
- DCL (Device Clear)
- SDC (Selected Device Clear)
- TCT (Take Control)
- PPD (Parallel Poll Disable)
- SPE (Serial Poll Enable)
- SPD (Serial Poll Disable)
- PPC (Parallel Poll Configure)
- PPE (Parallel Poll Enable)
- PPU (Parallel Poll Unconfigure)

(注)
- LADは リスナ・アドレス，必要により２次アドレスが続く
- TADは トーカ・アドレス，必要により２次アドレスが続く
- UNLは アンリスナ
- UNTは アントーカ
- CLAは コントローラのリスナ・アドレス

〈表３〉[2),(26] GPIBのファンクション

SHファンクション	DIOライン上のメッセージを確実に送信する機能．
AHファンクション	DIOライン上のメッセージを受信する機能． バスに接続されたSHファンクションと，AHファンクションとのハンドシェイク・シーケンスによって，メッセージの非同期転送が行われる．
TまたはTEファンクション	装置がトーカとしてアドレス指定されたとき，その他の装置へデータをインターフェースを通じて送る機能．TEファンクションは，２次アドレスにより拡張トーカとして機能する．
LまたはLEファンクション	装置がリスナとしてアドレス指定されたとき，その他の装置からデータをインターフェースを通して受け取る機能．LEファンクションは，２次アドレスにより拡張リスナとして機能する．
SRファンクション	インターフェースを管理するコントローラへ，非同期のサービスを要求する機能．コントローラのシリアル・ポーリングにより，装置はステータス・バイトを送出する．
RLファンクション	装置が，リモート動作かローカル動作かを選択する機能．
PPファンクション	装置が，コントローラのパラレル・ポール時に，トーカに指定されることなく，コントローラに対して１ビットのステータスを送出する機能．DIOライン１本に１装置を割り当てることで，一度に８台の装置まで可能． コントローラへのサービス要求は，SRQメッセージを用いたものと，パラレル・ポールによるものがあります．前者は，装置がサービス要求をしたい場合，非同期でコントローラに送信可能ですが，１本の信号ラインでOR接続されているため，どの装置から要求があったのか判断できないといった短所があります．後者は，サービス要求をしている装置を，同時に８台まで認識できますが，コントローラが必要と感じたときのみで，装置がサービス要求したいときには，サービス要求ができないといった欠点があります．システムにより使い分けが必要ですが，一般的にSRQメッセージを使用したものが多く見られます．
DCファンクション	装置の初期化をする機能．
DTファンクション	リスナに指定された装置を，測定開始(または動作開始)する機能．
Cファンクション	インターフェースを通じて，装置のアドレス，コマンド，その他の装置に送る機能．またどのデバイスがサービスを要求しているかのシリアル・ポール，パラレル・ポールを行う機能を持つコントローラとしての機能　などです

§3-2 GPIBコントロールLSIの使い方

里　和政/松井雅行/竹尾佳己

GPIBを制御するためのハードウェアは，複雑なうえソフトウェアの負担を考え合わせると，専用のLSIを使用するほうが簡単になります．

各メーカから専用のLSIが発表されており，その中でも代表的なLSIについて説明します．

μPD7210

μPD7210は，IEEE-488の規格のインターフェース機能をもっています．PC9801のGPIB用のコントローラでも使用されています．

● μPD7210の特性

このLSIは，トーカ，リスナ，コントローラの機能をプログラムによって制御することができます．また，データ転送に必要なハンドシェイクも自動的に行えます．これらの制御には，16個のライト/リード・レジスタを使用します．

図1にピン配置を，図2に内部ブロックを，図3にコントロール・レジスタを示します．図4に電気的特性およびタイムチャートを示します．

● 応用回路例

図5に，μPD7210を80系バスに接続するときの回路図を示します．この図のI/Oアドレスは，８０Ｈ～８７Ｈ番地までをコントローラに，８８Ｈまたは８９Ｈ番地をアドレス・スイッチ(S_1)データ・リード用に使用しています．また，DMAREQ端子を割り込み(DI/DO割り込み)に使用しています．

S_2スイッチを＋５Ｖ側にすることにより，システム・コントローラとして使用します．

◈ μPD7210のソフトウェア

GPIB用LSIを使用するためのソフトウェアは，電源投入時の初期設定，LSIから装置への割り込み要求，装置からLSIへの処理の三つに分類できます．

ここではμPD7210を使用してトーカ，リスナ機能を実現する場合を例にして説明します[12]．レジスタについては図3を参照してください．

● 初期設定

μPD7210は，電源投入時に補助コマンドのponメッセージによりpon状態にし，初期設定を行ってからpon状態を解除することにより，初めて使用可能となります．

初期設定ではつぎのことを行います．

① 割り込みマスク・レジスタの設定
② シリアル・ポール・モードのクリア
③ アドレス・モード・レジスタの設定
④ アドレス・レジスタの設定
⑤ クロック周波数の設定
⑥ データ受信モードの設定
⑦ T_1ディレイ時間の設定

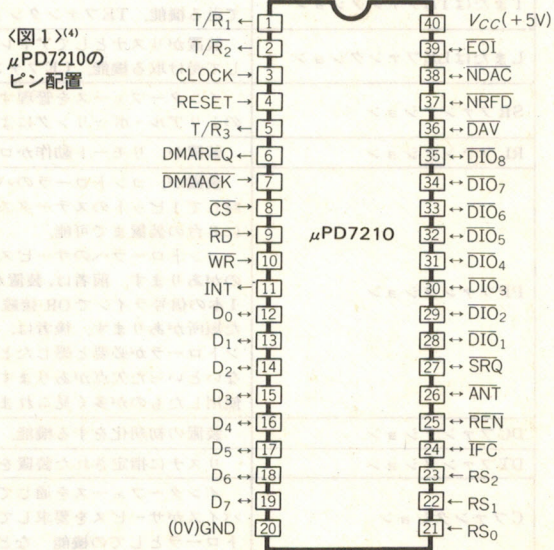
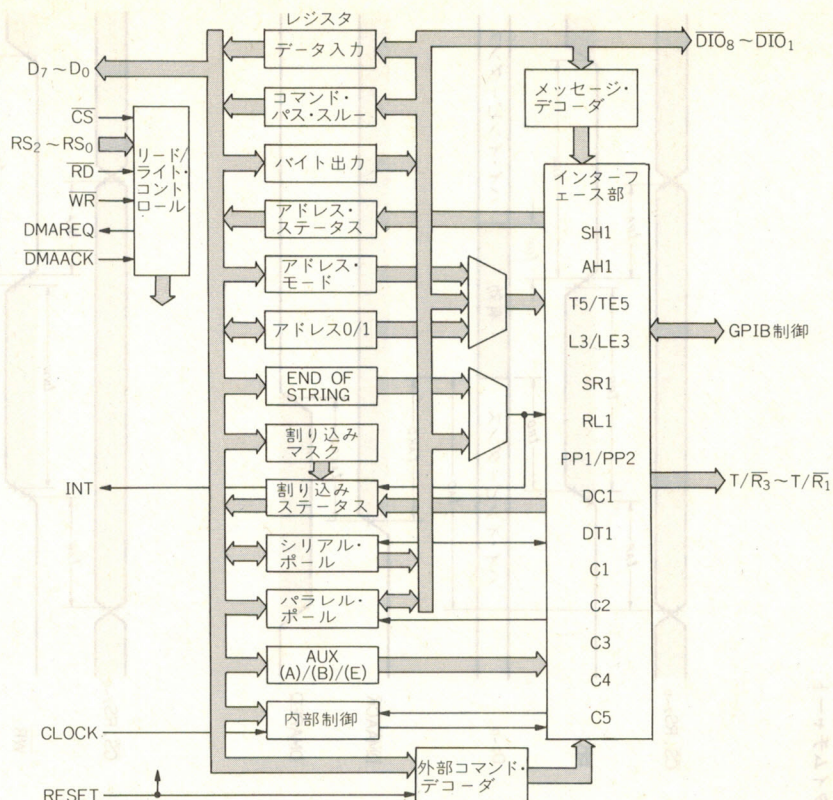
T_1ディレイ時間は，通常２μs以上です．ただし，$DIO_{1\sim8}$，DAV，EOIラインに３ステート・ドライバを使用している場合には1100ns以上で，かつ２バイト目以降は500ns以上でよいことになっています．

⑧ DC，DTファンクションのホールド・オフの設定

初期設定のフローチャートを図６に示します．

● 割り込み処理

μPD7210から装置への割り込み要求は13種類ありますが，初期設定ではつぎの７種類の割り込みをイネ

〈図1〉[4] μPD7210のピン配置

〈図２〉[4]
μPD7210の内部ブロック

〈図３〉[4] **μPD7210の内部レジスタ**

リード・レジスタ

	D7	D6	D5	D4	D3	D2	D1	D0	名称	RS$_2$RS$_1$RS$_0$
(0R)	DI$_7$	DI$_6$	DI$_5$	DI$_4$	DI$_3$	DI$_2$	DI$_1$	DI$_0$	データ入力	0 0 0
(1R)	CPT	APT	DET	END	DEC	ERR	DO	DI	割り込みステータス１	0 0 1
(2R)	INT	SRQI	LOK	REM	CO	LOKC	REMC	ADSC	割り込みステータス２	0 1 0
(3R)	S$_8$	PEND	S$_6$	S$_5$	S$_4$	S$_3$	S$_2$	S$_1$	シリアル・ポール・ステータス	0 1 1
(4R)	CIC	$\overline{\mathrm{ATN}}$	SPMS	LPAS	TPAS	LA	TA	MJMH	アドレス・ステータス	1 0 0
(5R)	CPT$_7$	CPT$_6$	CPT$_5$	CPT$_4$	CPT$_3$	CPT$_2$	CPT$_1$	CPT$_0$	コマンド・パス・スルー	1 0 1
(6R)	×	DT$_0$	DL$_0$	AD$_{5-0}$	AD$_{4-0}$	AD$_{3-0}$	AD$_{2-0}$	AD$_{1-0}$	アドレス０	1 1 0
(7R)	EOI	DT$_1$	DL$_1$	AD$_{5-1}$	AD$_{4-1}$	AD$_{3-1}$	AD$_{2-1}$	AD$_{1-1}$	アドレス１	1 1 1

ライト・レジスタ

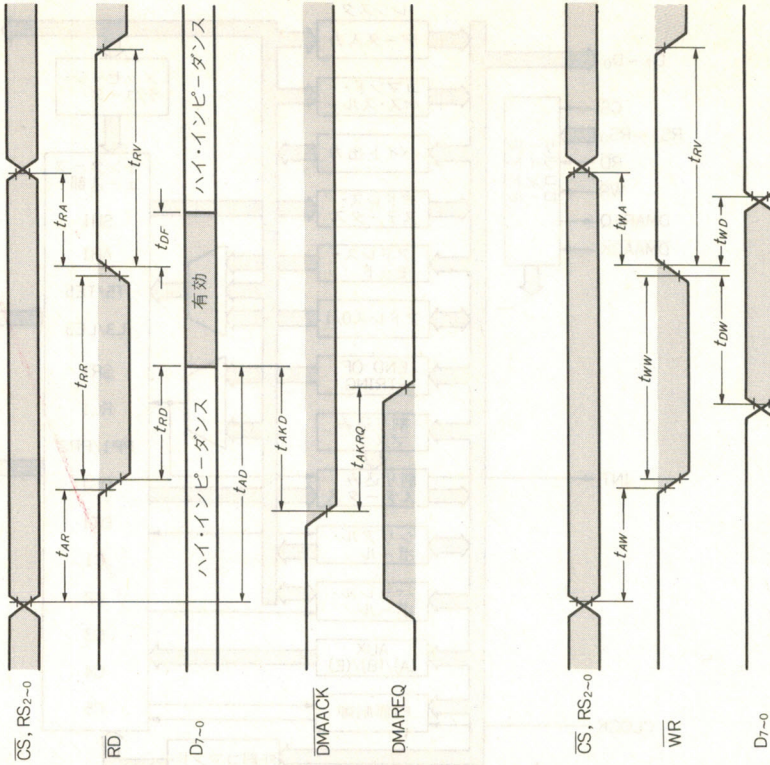
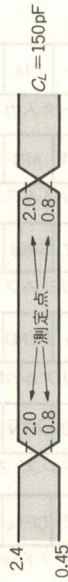
RS$_2$RS$_1$RS$_0$	D7	D6	D5	D4	D3	D2	D1	D0		名称
0 0 0	BO$_7$	BO$_6$	BO$_5$	BO$_4$	BO$_3$	BO$_2$	BO$_1$	BO$_0$	(0W)	バイト出力
0 0 1	CPT	APT	DET	END	DEC	ERR	DO	DI	(1W)	割り込みマスク１
0 1 0	0	SRQI	DMAO	DMAI	CO	LOKC	REMC	ADSC	(2W)	割り込みマスク２
0 1 1	S$_8$	rsv	S$_6$	S$_5$	S$_4$	S$_3$	S$_2$	S$_1$	(3W)	シリアル・ポール・モード
1 0 0	ton	lon	TRM$_1$	TRM$_0$	0	0	ADM$_1$	ADM$_0$	(4W)	アドレス・モード
1 0 1	CNT$_2$	CNT$_1$	CNT$_0$	COM$_4$	COM$_3$	COM$_2$	COM$_1$	COM$_0$	(5W)	AUX モード
1 1 0	ARS	DT	DL	AD$_5$	AD$_4$	AD$_3$	AD$_2$	AD$_1$	(6W)	アドレス０/１
1 1 1	EC$_7$	EC$_6$	EC$_5$	EC$_4$	EC$_3$	EC$_2$	EC$_1$	EC$_0$	(7W)	End of String

〈図 4 (a)〉[4] μPD7210のタイムチャート

項　　目	記号	条　　件	min	max
$\overline{\mathrm{EOI}}$↓→$\overline{\mathrm{DIO}}$ 遅延時間	t_{EODI}	PPSS→PPAS, ATN=True		250
$\overline{\mathrm{EOI}}$↓→T/$\mathrm{R_1}$↑ 遅延時間	t_{EOT11}	PPSS→PPAS, ATN=True		155
$\overline{\mathrm{EOI}}$↑→T/$\mathrm{R_1}$↓ 遅延時間	t_{EOT12}	PPAS→PPSS, ATN=False		200
$\overline{\mathrm{ATN}}$↓→$\overline{\mathrm{NDAC}}$↓ 遅延時間	t_{ATND}	AIDS→ANRS, LIDS		155
$\overline{\mathrm{ATN}}$↓→T/$\mathrm{R_1}$↓ 遅延時間	t_{ATT1}	TACS+SPAS→TADS, CIDS		155
$\overline{\mathrm{ATN}}$↓→T/$\mathrm{R_2}$↓ 遅延時間	t_{ATT2}	TACS+SPAS→TADS CIDS		200
$\overline{\mathrm{DAV}}$↓→DMAREQ↑ 遅延時間	t_{DVRQ}	ACRS→ACDS LACS		600
$\overline{\mathrm{DAV}}$↓→$\overline{\mathrm{NRFD}}$↓ 遅延時間	t_{DVNR1}	ACRS→ACDS		350
$\overline{\mathrm{DAV}}$↓→$\overline{\mathrm{NDAC}}$↑ 遅延時間	t_{DVND1}	ACRS→ACDS→AWNS		650
$\overline{\mathrm{DAV}}$↑→$\overline{\mathrm{NDAC}}$↓ 遅延時間	t_{DVND2}	AWNS→ANRS		350
$\overline{\mathrm{DAV}}$↑→$\overline{\mathrm{NRFD}}$↑ 遅延時間	t_{DVNR2}	AWNS→ANRS→ACRS		350
$\overline{\mathrm{RD}}$↓→$\overline{\mathrm{NRFD}}$↑ 遅延時間	t_{RNR}	ANRS→ACRS LACS, DI reg.selected		500
$\overline{\mathrm{NDAC}}$↑→DMAREQ↑ 遅延時間	t_{NDRQ}	STRS→SWNS→SGNS, TACS		400
$\overline{\mathrm{NDAC}}$↑→$\overline{\mathrm{DAV}}$↑ 遅延時間	t_{NDDV}	STRS→SWNS→SGNS		350
$\overline{\mathrm{WR}}$↑→$\overline{\mathrm{DIO}}$ 遅延時間	t_{WDI}	SGNS→SDYS, BO reg.selected		250
$\overline{\mathrm{NRFD}}$↑→$\overline{\mathrm{DAV}}$↓ 遅延時間	t_{NRDV}	SDYS→STRS, $\mathrm{T_1}$=True		350
$\overline{\mathrm{WR}}$↑→$\overline{\mathrm{DAV}}$↓ 遅延時間	t_{WDV}	SGNS→SDYS→STRS BO reg.selected, RFD=True $N_F=f_c=$ 8 MHz, $\mathrm{T_1}$(高速)		t_{SYNC} +830
TRIG パルス幅	t_{TT}		50	

$\overline{\mathrm{CS}}$, $\mathrm{RS_{2\sim0}}$
$\overline{\mathrm{RD}}$
$\mathrm{D_{7\sim0}}$
ハイ・インピーダンス　有効　ハイ・インピーダンス
$\overline{\mathrm{DMAACK}}$
DMAREQ
t_{AR}　t_{RR}　t_{RA}　t_{RV}　t_{RD}　t_{DF}　t_{AD}　t_{AKD}　t_{AKRQ}

$\overline{\mathrm{CS}}$, $\mathrm{RS_{2\sim0}}$
$\overline{\mathrm{WR}}$
$\mathrm{D_{7\sim0}}$
t_{AW}　t_{WW}　t_{WA}　t_{RV}　t_{DW}　t_{WD}

〈図4(b)〉[4] μPD7210の電気的特性

(a) 絶対最大定格（T_a=25℃）

項　　目	記　号	定　　格	単位
電源電圧	V_{CC}	−0.5～7.0	V
入力電圧	V_I	−0.5～7.0	V
出力電圧	V_O	−0.5～7.0	V
動作温度	T_{opt}	0～+70	℃
保存温度	T_{stg}	−65～+150	℃

(b) DC特性（T_a= 0℃～+70℃, V_{CC}=+5V±10%）

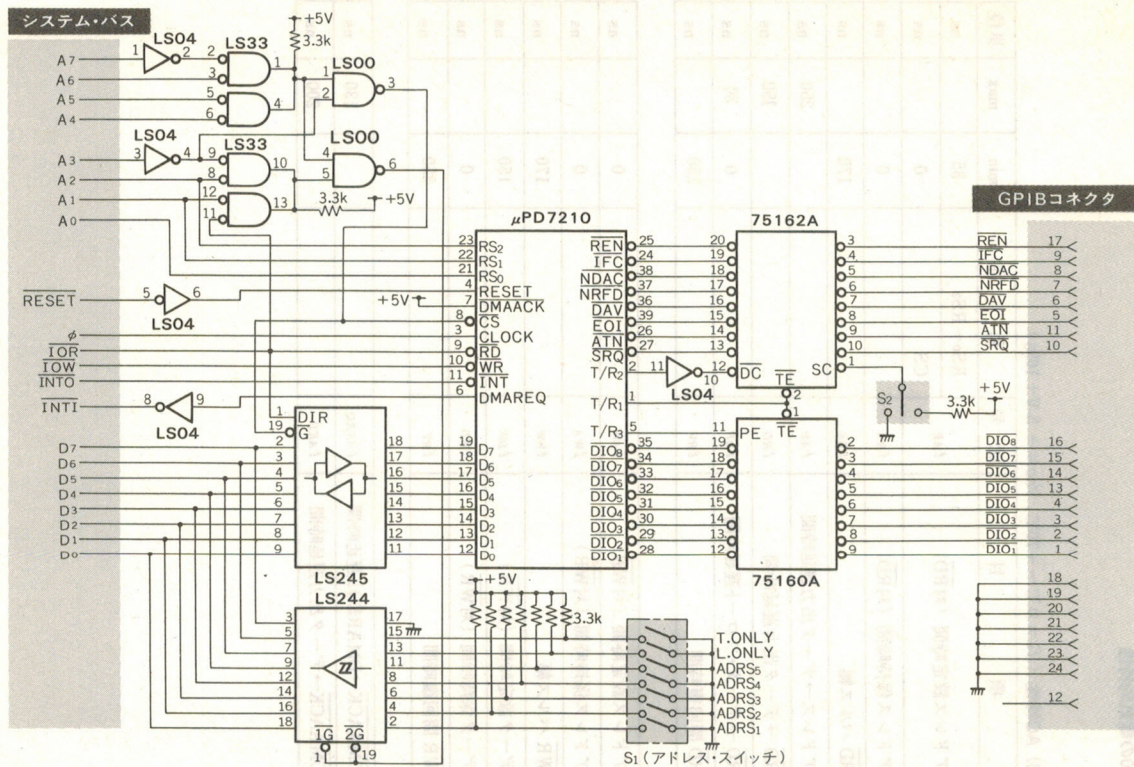
項　　目	記　号	条　　件	min	max	単位
低レベル入力電圧	V_{IL}		−0.5	0.8	V
高レベル入力電圧	V_{IH}		2.0	V_{CC}+0.5	V
低レベル出力電圧	V_{OL}	I_{OL}= 2mA, T/R1以外		0.45	V
		I_{OL}= 4mA, T/R1		0.45	V
高レベル出力電圧	V_{OH1}	I_{OH}=−400μA, INT以外	2.4		V
	V_{OH2}	I_{OH}=−400μA　INT	2.4		V
		I_{OH}=−50μA　INT	3.5		V
入力リーク電流	I_{LI}		−10	10	μA
出力リーク電流	I_{LO}		−10	10	μA
電源電流	I_{CC}			180	mA

(c) 容量（T_a=25℃, V_{CC}=GND=0V）

項　　目	記　号	条　　件	max	単位
入力容量	C_{IN}	f=1MHz 被測定端子以外は0V	10	pF
出力容量	C_{OUT}		15	pF
入出力容量	$C_{I/O}$		20	pF

(d) AC特性（T_a= 0℃～+70℃, V_{CC}=+5V±10%）

項　　目	記　号	条　　件	min	max	単位
アドレス設定時間（対$\overline{RD}$）	t_{AR}	RS0～RS2	85		ns
		CS	0		ns
アドレス保持時間（対$\overline{RD}$）	t_{RA}		0		ns
$\overline{RD}$ パルス幅	t_{RR}		170		ns
アドレス→データ出力遅延時間	t_{AD}			250	ns
$\overline{RD}$→データ出力遅延時間	t_{RD}			150	ns
$\overline{RD}$→データ・フロート遅延時間	t_{DF}		0	80	ns
$\overline{RD}$ 間回復時間	t_{RV}		250		ns
アドレス設定時間（対$\overline{WR}$）	t_{AW}		0		ns
アドレス保持時間（対$\overline{WR}$）	t_{WA}		0		ns
$\overline{WR}$ パルス幅	t_{WW}		170		ns
データ設定時間（対$\overline{WR}$）	t_{DW}		150		ns
データ保持時間（対$\overline{WR}$）	t_{WD}		0		ns
$\overline{WR}$ 間回復時間	t_{RV}		250		ns
$\overline{DMAACK}$→DMAREQ↓遅延時間	t_{AKRQ}			130	ns
$\overline{DMAACK}$→データ出力遅延時間	t_{AKD}			200	ns

〈図5〉[1] μPD7210の応用回路図

ーブルにしました．

① DI(Data In)

バイト入力レジスタにデータ・バイトが蓄えられていることを示しています．リスナに指定され，アクセプタ・ハンドシェイクを行おうとしています．

② DO(Data Out)

バイト出力レジスタへのデータ・バイトの書き込み要求，トーカに指定され，ソース・ハンドシェイクを行おうとしています．

③ DEC(Device Clear)

DCLまたはSDCコマンドを受信した．

④ DET(Device Trigger)

GETコマンドを受信した．

⑤ ADSC(Address Status Change)

トーカまたはリスナのアクティブ/アイドル状態が遷移した．

⑥ REMC(Remote Change)

リモート/ローカル状態が遷移した．

⑦ LOKC(Lockout Change)

リモート/ローカルのロック状態が遷移した．

割り込み処理のフローチャートを図7に示します．

● **装置の処理**

トーカ，リスナ機能では，装置からμPD7210への処理にはつぎの処理があります．

① データ受信

リスナに指定されたことにより，データの受信を行う．

② データ送信

トーカに指定されたことにより，データの送信を行う．

③ サービス要求

コントローラに対してサービスを要求する．

④ ローカル状態に設定

装置のローカル・スイッチが押されたことにより，LSIをローカル状態にする．

装置の処理のフローチャートを図8に示します．またプログラム例を**リスト1**(pp.92〜94)に示します．

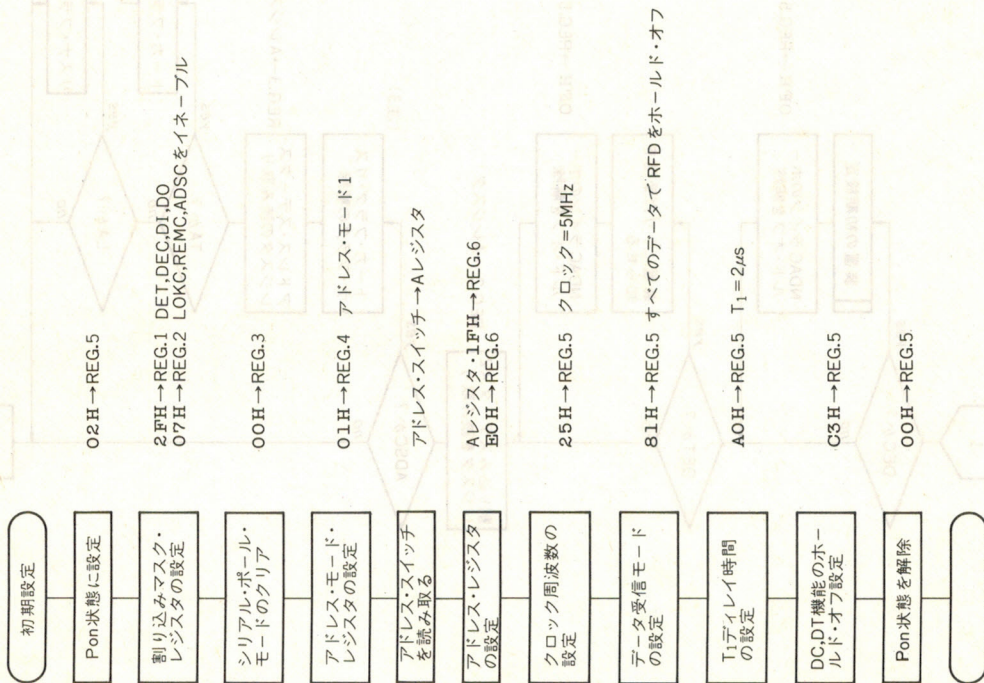
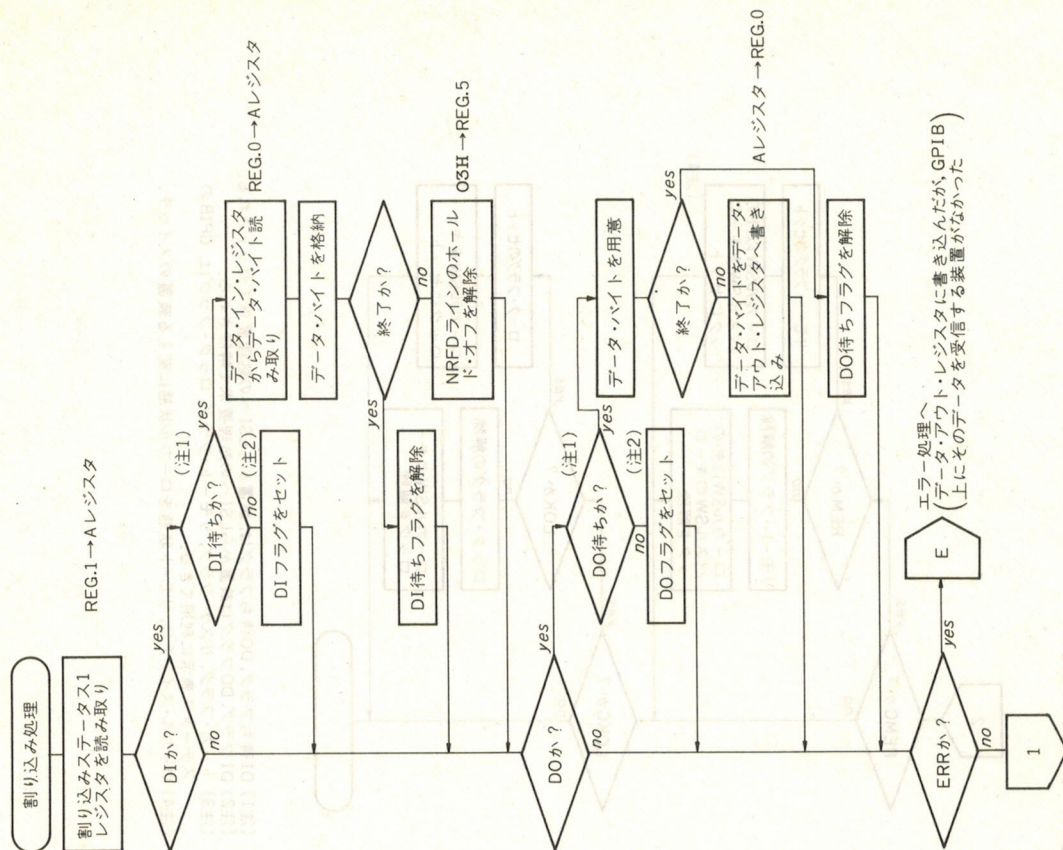
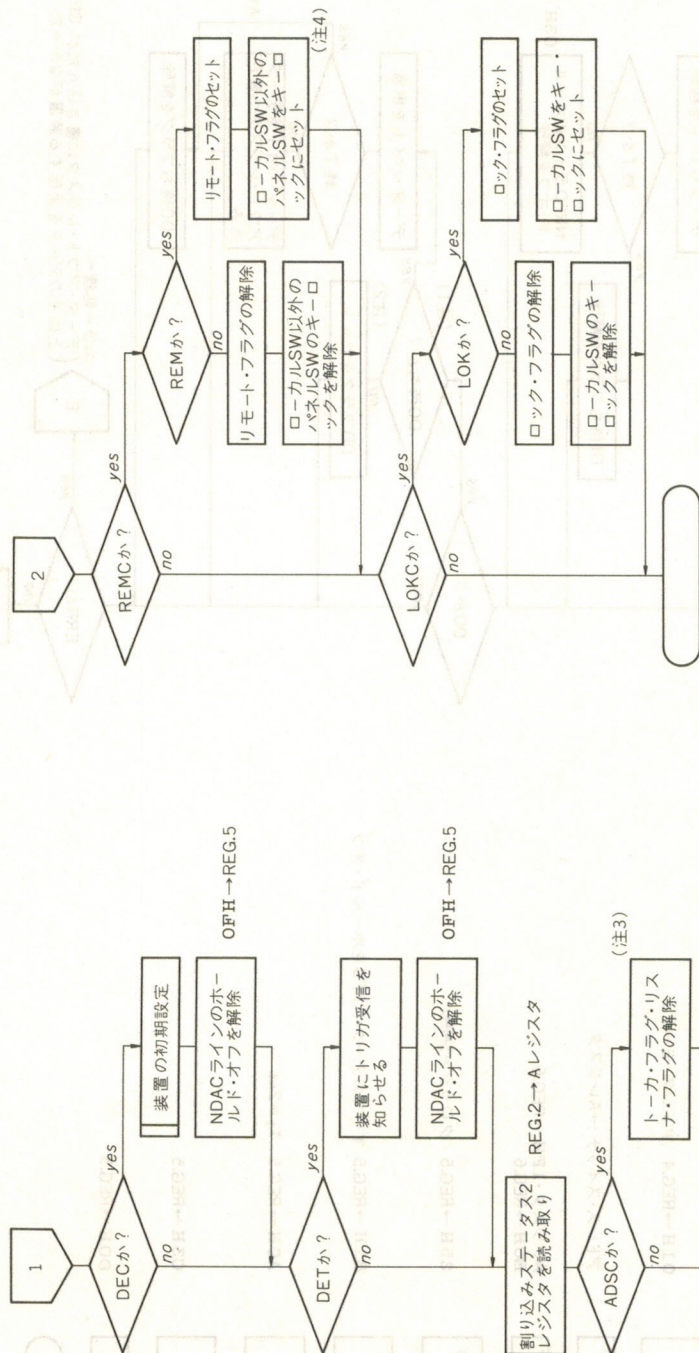
MC68488

MC68488は，68系のモトローラ社のGPIA(General Purpose Interface Adapter)で，かなり早い時期に発表されました．

● **MC68488の特徴**

このLSIは，トーカ，リスナの機能だけでコントローラとしての機能は，付加されていません．もっぱら計測器に組み込むためのものです．

そのためコントローラとして使用する場合は，

〈図6〉初期設定のフローチャート

〈図7〉割り込み処理フローチャート

〈図7〉割り込み処理フローチャート(つづき)

(注1) DI待ちフラグ,DO待ちフラグは,装置からLSIへの処理要求でセットされる.
(注2) DIフラグ,DOフラグは装置からLSIへの処理要求で解除される.
(注3) トーカ・フラグ,リスナ・フラグ,リモート・フラグ,ロック・フラグは,GPIBのステータス表示に利用できる.
(注4) ローカル・スイッチは,リモート状態をローカル状態に変える装置のスイッチ.

〈図8〉装置処理のフローチャート

(注1) データ受信では，リスナに指定されていることが条件になる．リスナに指定されていない場合，リスナに指定されるまで待つわけであるが，たんに待つか別の処理を行いながら待つかを判断する．

(注2) このループ中は，LSIからの割り込みにより，データ受信が行われている．DI待ちフラグが解除されたら，正常にデータ受信が終了したことになり，リスナが解除されたら，コントローラにより強制的に終了させられたことになる．

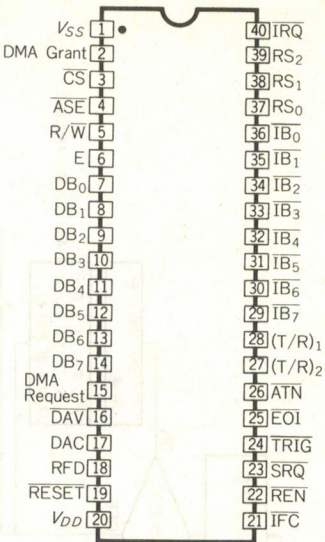
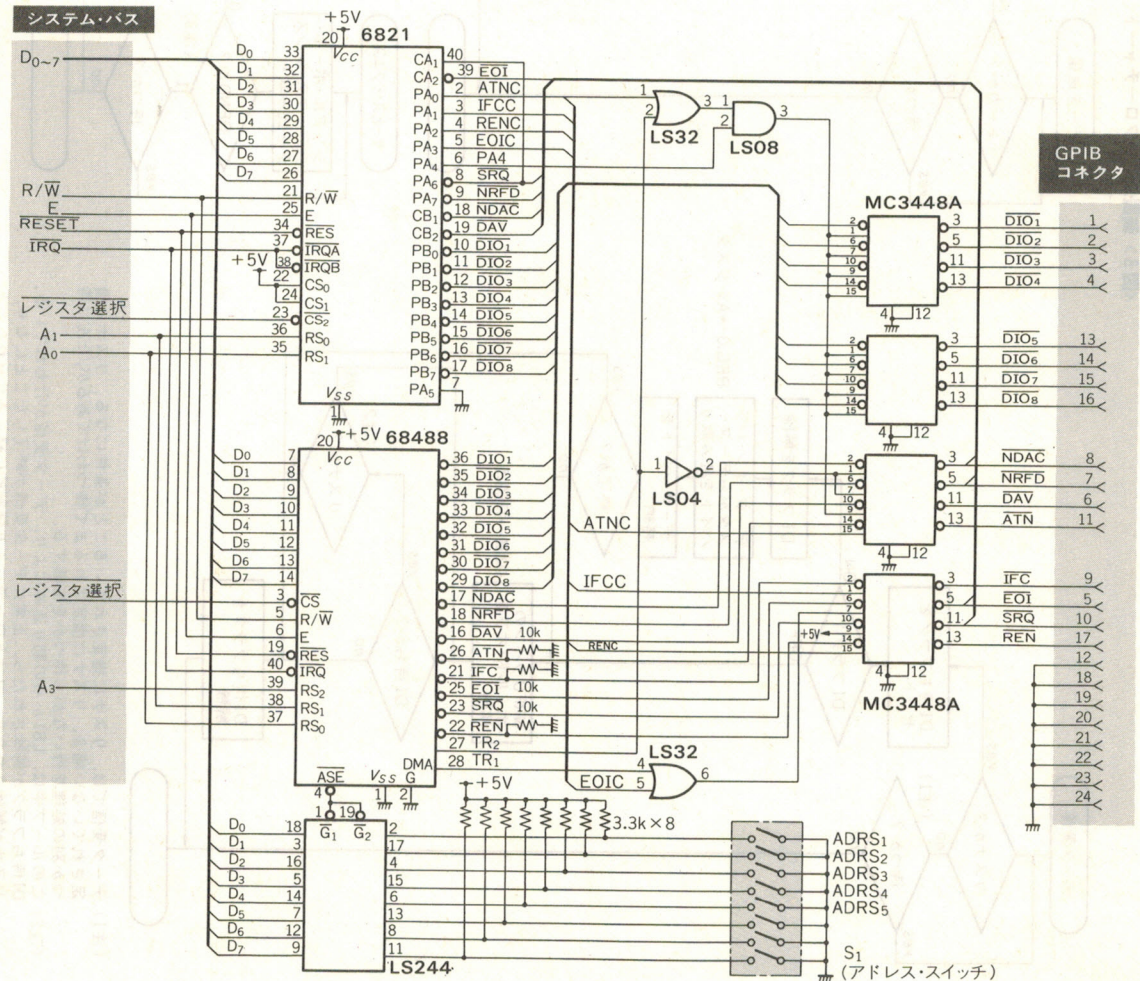
(注3) 以前に送ったサービス・リクエストがコントローラに受け付けられたかを調べる．もし，受け付けられていなければ，受け付けられるまで待つ．

ATN, IFC, EOI, SRQおよびREN信号線をハードを付加して制御する必要があります．簡単には，PIA（MC6821）を付加することで可能です．

GPIAの制御は，制御レジスタによって行い，リード・レジスタが8個，ライト・レジスタが7個あります．コマンドの実行結果は，割り込みレジスタ（R0R）にセットされます．そのために割り込みを使用しない場合は，このレジスタをポーリングして状態を確認するようにします．

図9にピン配置，**表1**に各ピンの説明，**表2**に内部レジスタ，**表3**にレジスタの説明をそれぞれ示します．

図10にMC68488の回路図を示します．PIAを使用してコントロールも可能にします．GPIAは，R4Rレジスタをリードすると$\overline{ASE}$端子から“L”パルスが出力されます．これによってアドレス・スイッチをリードすることができます．

〈図9〉[6] MC68488のピン配置

〈図10〉[1] MC68488の応用回路図

◈ MC68488のソフトウェア

GPIAの制御ソフトは，初期設定，トーカ，リスナおよびコントローラについて説明します．今回は，割り込みを使用していません．

● 初期設定

電源投入時にハード・リセットが行われますが，制御レジスタ(R3W)によってリセットします．以下処理手順を示します．

① アドレス・スイッチのリード
② アドレス・レジスタの設定
③ リセットのクリア
④ PIAの初期設定
⑤ トーカ，リスナの設定

〈表1〉(5) MC68488の各入出力ピン説明

ピン番号	名称	各端子の働き	ピン番号	名称	各端子の働き
1	V_{SS}	グラウンド端子	21	$\overline{IFC}$	GPIAを既知状態にするクリア端子
2	DMA Grant	DMA要求を知らせる入力端子 使用しない時は必ず接地	22	$\overline{REN}$	リモート"L"，ローカル"H"の選択端子
3	$\overline{CS}$	GPIAをセレクトするチップ・セレクト端子	23	$\overline{SRQ}$	サービス要求を出力する端子
4	$\overline{ASE}$	アドレス設定の時使用する端子	24	TRIG	IEEE-標準バスからのGETコマンドを知らせる端子
5	$R/\overline{W}$	レジスタのアクセスやデータの方向性を決めるリード・ライト端子	25	$\overline{EOI}$	データの出力転送の時，最後であることを知らせる端子
6	E	ϕ_2クロックの入力端子	26	$\overline{ATN}$	データ・バス上の信号がデータ"H"であるのかコマンドあるいはアドレス"L"であるかを示すアテンション端子
7	DB_0	MPUとGPIA間でのデータ伝送端子	27	$(T/\overline{R})_2$	トランシーバ(MC3448)のEOI以外のデータ伝送の方向性を制御する端子
8	DB_1	〃	28	$(T/\overline{R})_1$	トランシーバ(MC3448)のEOI信号の方向性を制御する端子
9	DB_2	〃	29	$\overline{IB_7}$	GPIBとGPIA間でのデータ伝送端子
10	DB_3	〃	30	$\overline{IB_6}$	〃
11	DB_4	〃	31	$\overline{IB_5}$	〃
12	DB_5	〃	32	$\overline{IB_4}$	〃
13	DB_6	〃	33	$\overline{IB_3}$	〃
14	DB_7	〃	34	$\overline{IB_2}$	〃
15	DMA Reguest	DMAコントローラに対し，DMAリクエストを出力する端子	35	$\overline{IB_1}$	〃
16	$\overline{DAV}$	ハンドシェイク・ラインのひとつで，トーカの際データが有効であることを示す	36	$\overline{IB_0}$	〃
17	DAC	ハンドシェイク・ラインのひとつで，リスナの際データの受け取り可能な時，真になる	37	RS_0	内蔵されている15個のレジスタのセレクト端子
18	RFD	ハンドシェイク・ラインのひとつで，リスナの際データを受け取った時，真になる	38	RS_1	〃
19	$\overline{RESET}$	MPUなどからハード的にGPIAをリセットする端子	39	RS_2	〃
20	V_{DD}	+5V端子	40	$\overline{IRQ}$	割り込み要求の出力端子

〈表2〉(5) MC68488内部レジスタ

レジスタ名称	名称	RS_2	RS_1	RS_0	$R/\overline{W}$	各レジスタの内容 7	6	5	4	3	2	1	0
割り込みステータス	R0R	0	0	0	1	INT	BO	GET	―	APT	CMD	END	BI
割り込みマスク	R0W	0	0	0	0	IRQ	BO	GET	―	APT	CMD	END	BI
コマンド・ステータス	R1R	0	0	1	1	UACG	REM	LOK	―	RLC	SPAS	DCAS	UUCG
(未使用)	―	0	0	1	0	―	―	―	―	―	―	―	―
アドレス・ステータス	R2R	0	1	0	1	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
アドレス・モード	R2W	0	1	0	0	dsel	to	lo	―	hlde	hlda	―	apte
補助コマンド	R3R	0	1	1	1	RESET	DAC	DAV	RFD	msa	rtl	ulpa	fget
補助コマンド	R3W	0	1	1	0	RESET	rfdr	feoi	dacr	msa	rtl	dacd	fget
アドレス・スイッチ	R4R	1	0	0	1	UD_3	UD_2	UD_1	AD_5	AD_4	AD_3	AD_2	AD_1
アドレス	R4W	1	0	0	0	lsbe	dal	dat	AD_5	AD_4	AD_3	AD_2	AD_1
シリアル・ポール	R5R	1	0	1	1	S_7	SRQS	S_5	S_4	S_3	S_2	S_1	S_0
シリアル・ポール	R5W	1	0	1	0	S_7	rsv	S_5	S_4	S_3	S_2	S_1	S_0
コマンド・パススルー	R6R	1	1	0	1	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
パラレル・ポール	R6W	1	1	0	0	PPR_8	PPR_7	PPR_6	PPR_5	PPR_4	PPR_3	PPR_2	PPR_1
データ・イン	R7R	1	1	1	1	DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0
データ・アウト	R7W	1	1	1	0	DO_7	DO_6	DO_5	DO_4	DO_3	DO_2	DO_1	DO_0

〈表3〉[5] MC68488の内部レジスタの説明

レジスタ名	ビット名	各ビットの働き
割り込みステータス・レジスタ(R0R)	BI	R7RにGPIBからのデータが読み込まれたかどうかの判断をする
	END	EOIが"L"でATNが"H"の時セットされる
	CMD	コントローラから特別なコマンドが送信された時セットされる
	APT	R6Rから第2アドレスが読み込まれた時セットされる
	GET	GETコマンドの判断をする
	BO	R7Wに書き込みが行われたかどうかを判断する
	INT	R0Wに書き込まれた内容により変化する
割り込みマスク・レジスタ(R0W)	BI	R0Rのそれぞれに対応する割り込みをマスクする
	END	
	CMD	
	APT	
	GET	
	BO	
	IRQ	IRQ端子からの割り込みをマスクする
コマンド・ステータス・レジスタ(R1R)	UUCG	コントローラから×001××××(×は1または0)コマンドが送信されるとセットされる
	DCAS	デバイス・クリア・アクティブ状態の時セットされる
	SPAS	シリアル・ポール・アクティブ状態の時セットされる
	RLC	リモート/ローカル状態に対応して変化する
	LOK	REN端子が"L"で，ローカル・ロックアウト・コマンド(×0010001)が送信されるとセットされる
	REM	リモート/ローカル状態を示すビット
	UACG	定義されていないアドレス・グループ・コマンドが送信されるとセットされる
アドレス・ステータス・レジスタ(R2R)	TPAS	第2アドレス・モード(apte＝1)の時セットされ，第1トーカ・アドレスを受信したことを示す
	LPAS	第2アドレス・モード(apte＝1)の時セットされ，第1リスナ・アドレスを受信したことを示す
	LACS	リスナ・アクティブ状態の時セットされる
	TACS	トーカ・アクティブ状態の時セットされる
	ATN	ATN端子の状態を示す
	lo	リスン・オンリ・モードの時セットされる
	to	トーク・オンリ・モードの時セットされる
	ma	TACS, LACS, SPASになった時などにセットされる
アドレス・モード・レジスタ(R2W)	apte	拡張アドレッシング・モードまたは第2アドレッシング・モードにする時にセットする
	hlda	リスナ・モードの時，RFDハンドシェイクをホールド・オフする時セット
	hlde	リスナ・モードでかつ，EOI信号を受け取った時，RFDハンドシェイクをホールド・オフする時セット
	lo	リスン・オンリ・モードにする時にセットする
	to	トーク・オンリ・モードにする時にセットする
	dsel	DCAS, UACG状態などにしない場合などにセットする
補助コマンド・レジスタ(R3R)	fget	TRIG端子の状態を知らせるビット
	ulpa	第1アドレスを受け取った時，最初のビットの内容を示す
	rtl	リモート状態からローカル状態になるとセットされる
	msa	第2アドレスが正しい存在状態の時セットされる
	RFD	RFD端子の状態を示すビット
	DAV	DAV端子の状態を示すビット
	DAC	DAC端子の状態を示すビット
	RESET	リセット状態を示すビット

レジスタ名	ビット名	各ビットの働き
補助コマンド・レジスタ(R3W)	fget	TRIG端子を"H"にする時セットする
	dacd	DACハンドシェイクを停止する時セットする
	rtl	セットするとリモートからローカル状態になる
	msa	第2コマンド・グループの伝送が行われている時セットする
	dacr	停止していたDACハンドシェイクを復帰する時セットする
	feoi	EOI信号を出力する時セット
	rfdr	RFDホールド・オフ・コマンドにより，停止していたハンドシェイクを復帰する時セット
	RESET	GPIAをリセットする時セットする
アドレス・スイッチ・レジスタ(R4R)	AD_1 〜 AD_5	GPIAのアドレス・スイッチのためのビット
	UD_1 〜 UD_3	ユーザが定義できるビット
アドレス・レジスタ(R4W)	AD_1 〜 AD_5	GPIAのアドレス書き込みのビット
	dat	トーカ機能を停止するビット
	dal	リスナ機能を停止するビット
	lsbe	デュアル・プライマリ・アドレス・モードにするビット
シリアル・ポール・レジスタ(R5R/W)	S_0 〜 S_5	シリアル・ポールを行う時のビット
	S_7	
	SRQS(R)	rsvに書き込んだ内容を示す
	rsv(W)	サービス要求する場合セット
コマンド・パススルー・レジスタ(R6R)	B_0 〜 B_7	IEEE-488標準バスのデータ・ラインの情報を示すビット
パラレル・ポール・レジスタ(R6W)	PPR_1 〜 PPR_8	パラレル・ポールを行うビット
データ・イン・レジスタ(R7R)	DI_0 〜 DI_7	GPIBからデータを受け取るビット
データ・アウト・レジスタ(R7W)	DO_0 〜 DO_7	MPUからデータをGPIBへ伝送するビット

(注) 各ビットの働きのすべてを記載したわけではない．
詳細は，GPIAユーザ・マニュアルを参照．

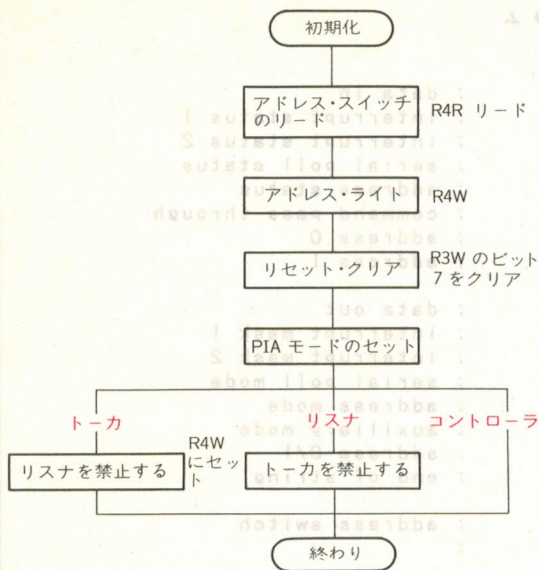
〈図11〉 MC68488初期化のフローチャート

などを行います。

図11に初期設定のフローチャートを示します。

● **トーカの処理**

トーカの場合は，データの送信を行います（図12のフローチャート参照）。

● **リスナの処理**

リスナの場合は，データの受信を行います（図13のフローチャート参照）。

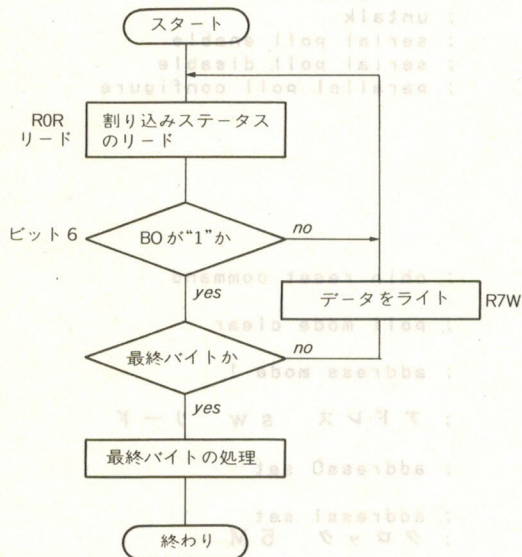
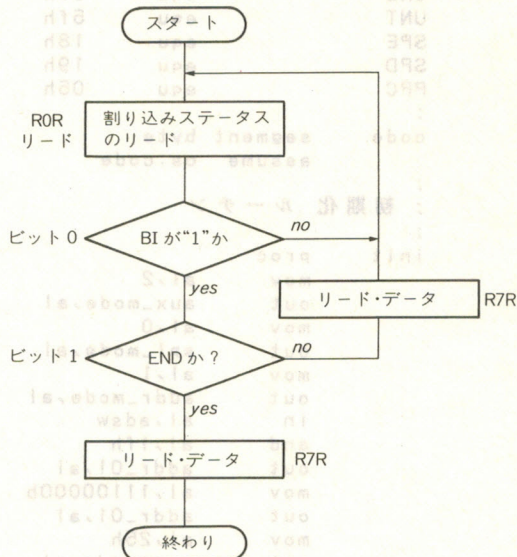
● **コントローラの処理**

コントローラの場合は，その機能によりトーカ，リスナに対してサービスを行います。割り込みにより，トーカ，リスナのデータ送受信の処理を行います。

GPIB上にメッセージを送る場合，ATN線によってコマンドまたはデータの区別を行います。

〈図12〉[6] トーカの処理

〈図13〉[6] リスナの処理

〈リスト1〉PC9801上のμPD7210によるGPIB入出力

```
;
;  GLIP  μPD7210  プログラム
;
;
data_in         equ     0               ; data in
int_sts1        equ     data_in+2       ; interrupt status 1
int_sts2        equ     int_sts1+2      ; interrupt status 2
spl_sts         equ     int_sts2+2      ; serial poll status
addr_sts        equ     spl_sts+2       ; address status
cmd_thr         equ     addr_sts+2      ; command pass through
addr0           equ     cmd_thr+2       ; address 0
addr1           equ     addr0+2         ; address 1
;
data_out        equ     data_in         ; data out
int_msk1        equ     int_sts1        ; interrupt mask 1
int_msk2        equ     int_sts2        ; interrupt mask 2
spl_mode        equ     spl_sts         ; serial poll mode
addr_mode       equ     addr_sts        ; address mode
aux_mode        equ     cmd_thr         ; auxiliary mode
addr_01         equ     addr0           ; address 0/1
end_str         equ     addr1           ; end of string
;
adsw            equ     addr1+2         ; address switch
cr              equ     0dh             ;
;
; command
;
UNL             equ     3fh             ; unlisten
UNT             equ     5fh             ; untalk
SPE             equ     18h             ; serial poll enable
SPD             equ     19h             ; serial poll disable
PPC             equ     05h             ; parallel poll configure
;
code    segment byte
        assume  cs:code
;
; 初期化 ルーチン
;
init    proc
        mov     al,2                    ; chip reset command
        out     aux_mode,al
        mov     al,0                    ; poll mode clear
        out     spl_mode,al
        mov     al,1                    ; address mode 1
        out     addr_mode,al
        in      al,adsw                 ; アドレス  sw  リード
        and     al,1fh
        out     addr_01,al              ; address0 set
        mov     al,11100000b
        out     addr_01,al              ; address1 set
        mov     al,25h                  ; クロック  5M
        out     aux_mode,al
        mov     al,81h                  ; RFD holdoff on all data mode
        out     aux_mode,al
        mov     al,0a0h                 ; T1=2μs
        out     aux_mode,al
        mov     al,0c3h                 ;
        out     aux_mode,al
        mov     al,0
        out     aux_mode,al             ; chip on
        ret
init    endp
;
gpib    proc
;
; コマンドの送信
;
; ES:BX=コマンドアドレス
```

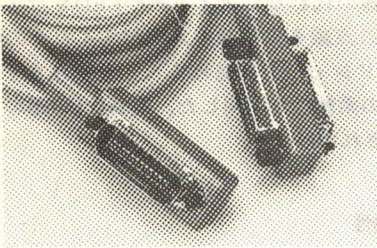
〈リスト1〉PC9801上のμPD7210によるGPIBの入出力(つづき)

```
; CX        =送信バイト数
;
cmd_send:
        call    set_casc                ; command mode set
cmd_out:
        mov     al,es:[bx]              ; command byte
        call    cmd_put                 ; command output
        loop    cmd_out
        ret
;
; データの送信
;
; ES:BX=データアドレス
; CX        =送信バイト数
;
data_send:
        call    set_casc                ; command mode set
        mov     al,10h                  ; go to standby command
        out     aux_mode,al             ; data send mode
data_set:
        mov     ah,es:[bx]              ; command byte
        call    data_put                ; command output
        loop    data_set
        ret
;
; データの受信
;
; ES:BX=データアドレス
; CX        =受信バイト数
;
data_recv:
        call    recv_data               ; data input
        mov     es:[bx],al
        inc     cx
        cmp     al,CR                   ; end code ?
        je      recv_end
        in      al,addr1
        test    al,80h                  ; EOI ?
        jz      data_recv
recv_end:
        ret
;
; シリアルポール
;
; DL=トーカアドレス
;
sril_poll:
        call    set_casc                ; command mode
        mov     ah,UNL                  ; Unlisten cmd.
        call    cmd_put
        mov     ah,SPE                  ; serial poll enable cmd.
        call    cmd_put
next_poll:
        mov     ah,dl                   ; トーカアドレス  send
        call    cmd_put
        mov     al,10h
        out     aux_mode,al             ; go to standby cmd.
        call    recv_data               ; STB read
        push    ax
        call    set_casc                ; command mode
        pop     ax
        test    al,40h                  ; REQ on ?
        jz      next_poll
        push    ax
        mov     ah,SPD
        call    cmd_put                 ; serial poll disable
```

〈リスト1〉PC9801上のμPD7210によるGPIBの入出力(つづき)

```
          mov     ah,UNT
          call    cmd_put                    ; untalk cmd.
          pop     ax
          ret
;
; パラレルポール
;
; ＤＬ＝リスナアドレス
; ＤＨ＝２次アドレスＰＰＥまたはＰＰＤ
;
prl_poll:
          call    set_casc
          mov     ah,UNL
          call    cmd_put                    ; UNL cmd send
          mov     ah,dl
          call    cmd_put                    ; リスナアドレス
          mov     ah,PPC
          call    cmd_put                    ; PPC send
          mov     ah,dh
          call    cmd_put                    ; PPE or PPD
          mov     al,1dh
          out     aux_mode,al                ; execute parallel poll
poll_chk:
          in      al,int_sts2
          test    al,10h                     ; co=1
          jz      poll_chk                   ; no
          in      al,cmd_thr                 ; poll response get
          push    ax
          mov     ah,UNL
          call    cmd_put
          pop     ax
          ret
; command send mode
;
set_casc:
          in      al,addr_sts                ; address status read
          test    al,40h                     ; ATN ?
          jnz     casc_exit
          mov     al,12h                     ; take control sync command
          out     aux_mode,al                ;
casc_exit:
          ret
;
cmd_put:
          in      al,int_sts2                ; interrupt status 2
          test    al,8h                      ; command out ?
          jz      cmd_put                    ; no
          mov     al,ah                      ;
          out     data_out,al                ; command out
          ret
;
data_put:
          in      al,int_sts2                ; interrupt status 2
          test    al,2h                      ; data out ?
          jz      data_put                   ; no
          mov     al,ah                      ;
          out     data_out,al                ; data out
          ret
;
recv_data:
          in      al,int_sts2
          test    al,1                       ; data in ?
          jz      recv_data                  ; no
          in      al,data_in
          ret
gpib      endp
code      ends
          end
```

GPIBラインを光ファイバで延長する

鶴野和孝

最近，FA化の波にのって，GPIBを標準またはオプションで装備した計測器やパソコンが多くなりました．

一方，オプト・エレクトロニクスの発達により，ゲートICなみに簡単に取り扱える光ファイバ・リンク・モジュールが各社より発表されています．この光ファイバ・リンクを利用することにより，

① 電磁誘導ノイズを受けない，あるいは出さない．
② 装置間がアイソレートされるので，短絡の危険がない．

などの利点が得られ，これらをGPIBと組み合わせて考えると，

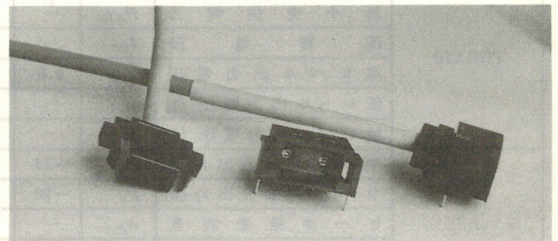
① GPIBを規格(最大20m)以上に延長できる．
② アナログ系に混入するノイズを低減できる．
③ 機器の故障がほかへ波及しない．

などの効果をあげられます．

そこで，ここでは広い工場や研究所などで計測システムを構築することを念頭におき，GPIBを規格以上に延長するという点について，製作実験を行った結果を報告したいと思います．

◆ 光ファイバ・リンク

各社より発表されている光ファイバ・リンクの中から，必要な周辺回路までがモールド・パッケージ化されているTOML75(東芝)を使用しました．

光ファイバ・リンクTOML75[13]は，**写真1**のように光送受信モジュールTODX75，光コネクタTOCP75，プラスチック光ファイバ(石英製もある)から構成されています．それぞれの電気的，光学的特性を**表1**に示

〈写真1〉光モジュールとコネクタ

〈表1(a)〉
トスリンクの絶対最大定格

型名	項目	記号	最小	最大	単位
TOML70/TOML75	保存温度	T_{stg}	−40	75	℃
	動作温度	T_{opr}	0	70	℃
TOTX70	供給電圧	V_{CC}	−0.5	7	V
	入力電圧	V_{IN}	−0.5	5.5	V
TORX70	供給電圧	V_{CC}	−0.5	7	V
	低レベル出力電流	I_{OL}	—	20	mA
	高レベル出力電流	I_{OH}	—	−1	mA
TODX75	供給電圧	V_{CC}	−0.5	7	V
	入力電圧	V_{IN}	−0.5	5.5	V
	低レベル出力電流	I_{OL}	—	20	mA
	高レベル出力電流	I_{OH}	—	−1	mA
TOCP70，TOCP70P TOCP75，TOCP75P	張力 光コネクタ/光ファイバ	T_{CF}	—	5	N
	張力 光ファイバ	T_F	—	50(100)[(1)]	N
	光ファイバ曲げ半径	r	15	—	mm
TOCP70Q，TOCP70R，TOCP70S TOCP75Q，TOCP75R，TOCP75S	張力 光コネクタ/光ファイバ	T_{CF}	—	50	N
	張力 光ファイバ	T_F	—	250(1000)[(2)]	N
	光ファイバ曲げ半径	r	50	—	mm

注(1) (　)内はTOCP70P，TOCP75Pの値
注(2) (　)内はTOCP75R，TOCP75Sの値

します．

バス延長装置の設計

GPIBでは，とくに各機器の応答速度を制限していません．したがって，§3-1の図3（p.77）におけるⒶ～Ⓕまでのどの区間をも引き延ばすことができます．例えば，機器の代わりにスイッチやLEDをバスに接続し，LEDの点滅を見ながら人間がスイッチのON/OFFでハンドシェイクを進めることもできます．この応答速度の柔軟性を利用した，簡易なGPIBのデバッガも発表されています．

〈表1(b)〉トスリンクの電気的/光学的特性

（T_{opr}＝25℃，V_{CC}＝5±0.25V，λ_P＝660nm）

型名	項目	記号	最小	標準	最大	単位	条件
TOML70 / TOML75	伝送容量		DC	—		Mbit/sec	任意符号(1)
	伝搬遅延時間（"L"→"H"）	t_{PLH}	—	175	250	ns	ファイバ長1m，Tx入力とRx出力間
	伝搬遅延時間（"H"→"L"）	t_{PHL}	—	175	250	ns	ファイバ長1m，Tx入力とRx出力間
TOTX70	ファイバ結合光出力(1)	P_f	−15	−13	−10	dBm	外付け抵抗150Ω，TOCP70-1MBを介して(2)
	ファイバ結合光出力(2)	P_f	−20	−17	−14	dBm	外付け抵抗150Ω，TOCP70Q-1MBを介して(2)
	ピーク発光波長	λ_P	—	660	—	nm	
	消費電流	I_{CC}	—	45	65	mA	
	高レベル入力電圧	V_{IH}	2.0	—	—	V	(3)
	低レベル入力電圧	V_{IL}	—	—	0.8	V	(3)
	高レベル入力電流	I_{IH}	—	—	40	μA	V_{CC}＝5.25V，V_{IH}＝2.4V
	低レベル入力電流	I_{IL}	—	—	−1.6	mA	V_{CC}＝5.25V，V_{IL}＝0.4V
TORX70	最大受信電力	P_{MAX}	−15	−13	—	dBm	BER＝10^{-9}(2)
	最小受信電力	P_{MIN}	—	−30	−28	dBm	BER＝10^{-9}(2)
	消費電流	I_{CC}	—	18	30	mA	
	高レベル出力電圧	V_{OH}	4.6	—	—	V	V_{CC}＝4.75V，I_{OH}＝−60μA(5)，−50μA(4)(6)
	低レベル出力電圧(1)	V_{OL}	—	—	0.4	V	V_{CC}＝4.75V，I_{OL}＝1.2mA(4)(5)
	低レベル出力電圧(2)	V_{OL}	—	—	0.5	V	V_{CC}＝4.75V，I_{OL}＝2mA(4)(6)
TODX75	ファイバ結合光出力(1)	P_f	−15	−13	−10	dBm	外付け抵抗150Ω，TOPC75-1MBを介して
	ファイバ結合光出力(2)	P_f	−20	−17	−14	dBm	外付け抵抗150Ω，TOPC75Q-1MBを介して
	ピーク発光波長	λ_P	—	660	—	nm	
	高レベル入力電圧	V_{IH}	2.0	—			(3)
	低レベル入力電圧	V_{IL}	—	—	0.8	V	(3)
	高レベル入力電流	I_{IH}	—	—	40	μA	V_{CC}＝5.25V，V_{IH}＝2.4V
	低レベル入力電流	I_{IL}	—	—	−16	mA	V_{CC}＝5.25V，V_{IL}＝0.4V
	最大受信電力	P_{MAX}	−15	−13	—	dBm	BER＝10^{-9}(2)
	最小受信電力	P_{MIN}	—	−30	−28	dBm	BER＝10^{-9}(2)
	消費電流	I_{CC}	—	63	95	mA	
	高レベル出力電圧	V_{OH}	4.6	—	—	V	V_{CC}＝4.75V，I_{OH}＝−60μA(5)，−50μA(4)(6)
	低レベル出力電圧(1)	V_{OL}	—	—	0.4	V	V_{CC}＝4.75V，I_{OL}＝1.2mA(4)(5)
	低レベル出力電圧(2)	V_{OL}	—	—	0.5	V	V_{CC}＝4.75V，I_{OL}＝2mA(4)(6)
プラスチック・ファイバ	開口数	NA	—	0.5	—		
	屈折率分布	ステップ・インデックス					
	伝送損失		—	0.4	—	dB/m	ファイバ長10mで測定
	帯域幅		—	60	—	MHz	ファイバ長50m
	伝搬遅延時間	t_P	—	5	7	ns/m	
ポリマ・クラッド石英コア・ファイバ	開口数	NA	—	0.41	—		
	屈折率分布	ステップ・インデックス					
	伝送損失		—	0.025	—	dB/m	ファイバ長100mで測定
	帯域幅		—	40	—	MHz	ファイバ長500m
	伝搬遅延時間	t_P	—	5	7	ns/m	

注(1) NRZ符号の場合は3Mbit/secまで動作
注(2) ピーク値で規定
注(3) 高レベル入力時：光出力OFF，低レベル入力時：光出力ON
注(4) 光入力OFF時：高レベル出力，光入力ON時：低レベル出力
注(5) LS TTLを3個接続した場合
注(6) S TTLを1個接続した場合

ところで，GPIBを利用したシステムを構築する場合，１台の機器でも20m以上の距離が離れていると規格を満足できません．

図１で示すような接続を行った場合，転送データの信頼性から装置間のケーブルの長さは，２m以下にする必要があります．またGPIBの負荷は，図２に示すように，３kと6.2kΩの分散方式となっているために，伝送速度には限界があります．またドライブ能力も48mAであるために，ケーブルを長くすることはノイズ・マージンを低下させます．そこで，基本的に図３のような場合を想定し，GPIBの延長装置を製作することにしました．

● GPIB延長装置の構成

GPIB延長装置の構成としてなるべく小型にするため，GPIBとのバッファ，光ファイバ・リンク・モジュール，UART，そしてZ80 CPUとメモリという構成にし，ソフトウェアによるハンドシェイクに依存することにしました．

図４にシステムのブロック図を示します．

● ATN応答対策

規格によれば，「ATN」がONした場合，すべての機器はそれまでのハンドシェイクを中断し，200ns以内にそれに応答しなくてはなりません．これはソフトウェアでは追いつけませんので，図５のようにハードウェアでカバーします．ここでは，装置自身が出力していないのに「ATN」がONになればフリップフロップをセットし，「NRFD」をONにします．

ソフトウェアのほうは，ポーリングによって「ATN」のONを検出した後，「NRFD」と「NDAC」をONにします．そして，次に「NRFD」のみOFFにすると，同時に「ATN」ONをもう一方の延長装置に光信号で送ります．

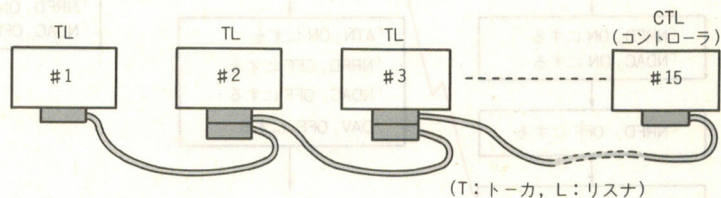
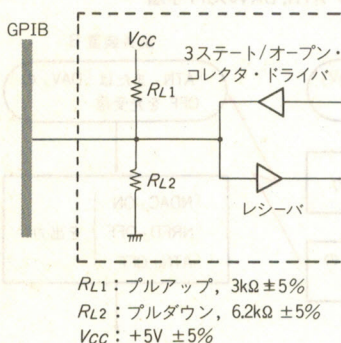
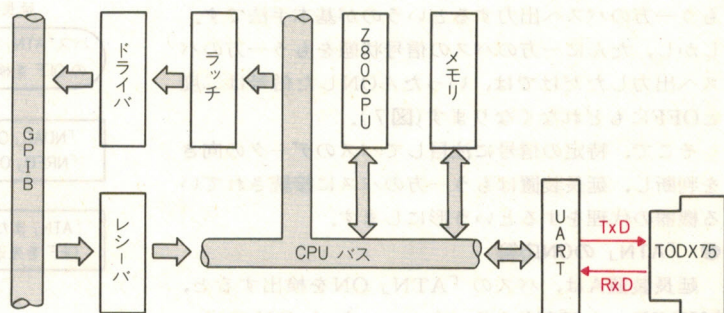
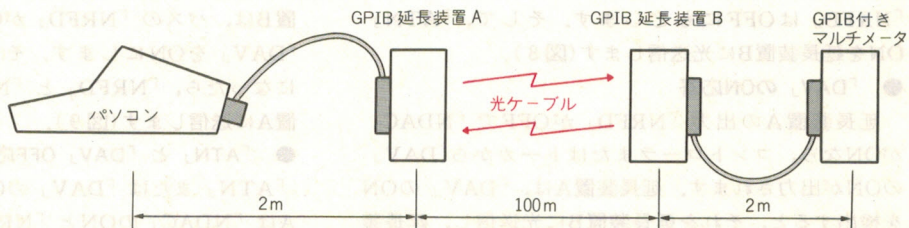
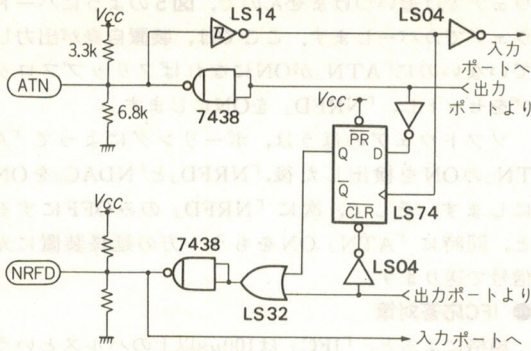
● IFC応答対策

規格によると，「IFC」は100μs以上のパルスということになっています．そこで，これもいったんフリップフロップで受け，ポーリングで検出した後(図６)，もう一方の延長装置に光信号で送ります．

IFCの光信号を受信した延長装置では，ワンショットをトリガして，GPIB上に100μs以上のパルスを送出するようにします．

ソフトウェア

このGPIBの延長装置は，一方のバスの信号状態を，

〈図１〉GPIBの接続例

〈図２〉GPIBの入出力部の特性

〈図４〉GPIB延長装置のブロック図

〈図３〉GPIBインターフェースで20m以上延長する例

〈図5〉ATNのタイミング発生回路

〈図7〉信号の送受信の注意点

〈図8〉ATNのON手順

〈図6〉IFCのタイミング発生回路

〈図9〉DAVのON手順

〈図10〉ATN, DAVのOFF手順

もう一方のバスへ出力するというのが基本手法です．しかし，たんに一方のバスの信号状態をもう一方のバスへ出力しただけでは，いったんONした信号は二度とOFFにもどれなくなります(図7)．

そこで，特定の信号に注目してバスのデータの向きを判断し，延長装置はもう一方のバスに接続されている機器の代理をするという形にします．

●「ATN」のON応答

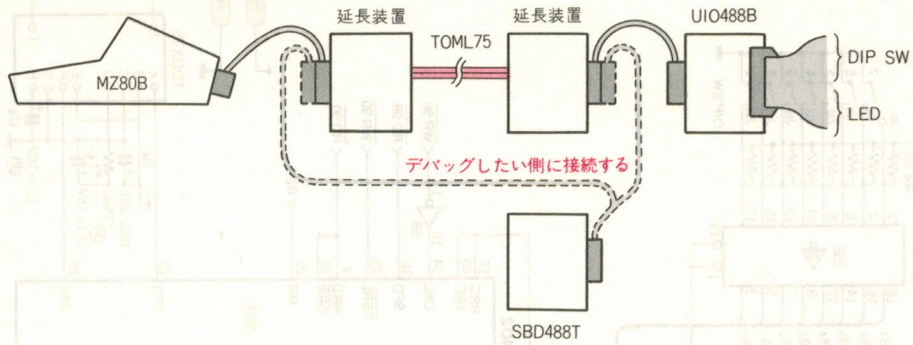
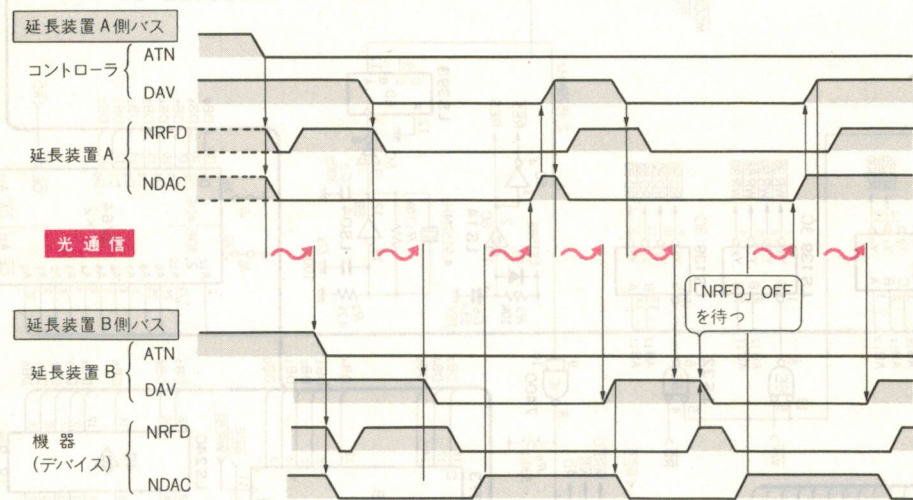
延長装置Aは，バスの「ATN」ONを検出すると，「NRFD」と「NDAC」をいったんONにし，「NRFD」はOFFにもどします．そして，「ATN」ONを延長装置Bに光送信します(図8)．

●「DAV」のON応答

延長装置Aの出力「NRFD」がOFFで「NDAC」がONなら，コントローラまたはトーカから「DAV」のONが出力されます．延長装置Aは，「DAV」のONを検出すると，それを延長装置Bに光送信し，延長装置Bは，バスの「NRFD」がOFFになるのを待って，「DAV」をONにします．そして，「NDAC」がOFFになったら，「NRFD」と「NDAC」の状態を延長装置Aに送信します(図9)．

●「ATN」と「DAV」OFF応答

「ATN」または「DAV」のOFFに対して，延長装置Aは「NDAC」のONと「NRFD」のOFFを出力し，

〈図11〉動作テストのブロック図

〈図13〉動作タイミング

次の「DAV」のONに備えます．また，「ATN」や「DAV」のOFFを延長装置Bに光送信します．

延長装置Bは，光受信すると「ATN」や「DAV」をOFFにし，「NDAC」のONと「NRFD」のOFFを出力します．この状態は，延長装置A，Bともに同じ出力です（図10）．

動作テスト

さて，実験に使用した装置は図11に示すように，コントローラとしてMZ80B（シャープ），機器（デバイス）としてUIO488B（マイクロサイエンス），さらにデバッグ用にSBD488T（マイクロサイエンス）を使用しました．

UIO488Bは，端末用のGPIBインターフェース・ユニットで，入出力ポート各1個を備えています．また，このUIO488Bが，リスナに指定されているか，トーカに指定されているかがわかるようになっているので，これらのピンにLEDやディップ・スイッチを接続し，動作が目で確認できるようにしました．

● 伝送速度

MZ80Bの転送速度は遅いので，ここではUIO488Bを2台使用し，1台をトーカ・オンリ，もう1台をリスナ・オンリにして，延長装置を通して実験しました．オシロスコープで波形を観測したところ，「DAV」と「DAV」の間隔が約5msあり，延長装置を通さない場合は約4μsでしたので，延長装置の伝送速度は約5ms/バイト（＝200バイト/秒）とみてよいと思います．

200バイト/秒というと相当に遅い速度ですが，1回当たりの転送データ数が少ない場合や，温度や真空などの信号のように，フィードバック応答の遅いシステムには，十分に応用できると思います．

また，何にも増して，GPIBを備えた機器に何の改造もせず遠隔操作できるメリットは大きいと思います．

最後に，この装置の回路図を図12に，タイミングを図13に，プログラムをリスト1に示します．

〈図12〉光インターフェース・システムの回路図

〈リスト1〉コントロール・プログラム

```
0000 =          ROMTOP: EQU     00000H
2000 =          RAMTOP: EQU     02000H
8000 =          ODB:    EQU     08000H  ;ORIGINAL DATA BUS PORT
8800 =          OCB:    EQU     08800H  ;ORIGINAL COMMAND BUS PORT
9000 =          SIOD:   EQU     09000H  ;SIO DATA PORT
9800 =          SIOC:   EQU     09800H  ;SIO COMMAND/STATUS PORT
A000 =          INTP:   EQU     0A000H
C000 =          DIPSW:  EQU     0C000H
E000 =          IFC:    EQU     0E000H  ;INTER FACE CLEAR PORT
                ;
099A =          CLK:    EQU     2458    ;KHZ (CPU CLOCK)
                ;
                ;
2000                    ORG     RAMTOP
2000            TSTMR:  DS      1
2001            RCTMR:  DS      1
2002            OCIBF:  DS      1       ;ORIGINAL COMMAND INPUT BF.
2003            OCOBF:  DS      1
2004            ODOBF:  DS      1
2005            LOCIBF: DS      1
2006            SDIBF:  DS      1
2007            SCIBF:  DS      1
2008            SCOBF:  DS      1
2009            SDOBF:  DS      1
200A            SNDBF:  DS      1
200B            SAREA:  DS      64
204B =          STACK:  EQU     $
                ;
                ;
0000                    ORG     ROMTOP
                ;
                ;
                ;
                ;
                INIT:   ;
0000 314B20             LXI     SP,STACK
0003 3E6C               MVI     A,6CH   ;DATA 8BITS,EVEN,2 STOP BITS
0005 320098             STA     SIOC
0008 3A0098             LDA     SIOC
000B 3A0090             LDA     SIOD
000E 3EFF               MVI     A,0FFH
0010 320020             STA     TSTMR
0013 320120             STA     RCTMR
0016 3E00               MVI     A,0
0018 320A20             STA     SNDBF
001B 320220             STA     OCIBF
001E 320320             STA     OCOBF
0021 320420             STA     ODOBF
0024 320620             STA     SDIBF
0027 320520             STA     LOCIBF
002A 320720             STA     SCIBF
002D 320820             STA     SCOBF
0030 320920             STA     SDOBF
                        ;
                MAIN:   ;
0033 3A0A20             LDA     SNDBF
0036 FE00               CPI     0
0038 C28401             JNZ     SEND
                        ;
003B 3A0220             LDA     OCIBF
003E 320520             STA     LOCIBF
0041 47                 MOV     B,A     ;SAVE LAST OCB
                        ;
0042 3A0080             LDA     ODB
0045 320920             STA     SDOBF
0048 3A0088             LDA     OCB
004B 320220             STA     OCIBF
004E 4F                 MOV     C,A     ;SAVE NOW OCB
                        ;
004F A8                 XRA     B
0050 5F                 MOV     E,A         ;SAVE ON/OFF EDGE
                        ;
0051 A1                 ANA     C
0052 6F                 MOV     L,A     ;SAVE ON EDGE
                        ;
0053 7B                 MOV     A,E
0054 A0                 ANA     B
0055 67                 MOV     H,A         ;SAVE OFF EDGE
                        ;
0056 79                 MOV     A,C
0057 B0                 ORA     B
0058 57                 MOV     D,A
0059 3A0320             LDA     OCOBF
005C AA                 XRA     D
005D 57                 MOV     D,A     ;SAVE BUS EDGE
                        ;
005E A5                 ANA     L
005F 6F                 MOV     L,A
                        ;
0060 7A                 MOV     A,D
0061 A4                 ANA     H
0062 67                 MOV     H,A
                        ;
0063 3A0820             LDA     SCOBF
0066 B5                 ORA     L
0067 47                 MOV     B,A
0068 7C                 MOV     A,H
0069 2F                 CMA
006A A0                 ANA     B
006B 320820             STA     SCOBF
                        ;
006E E604               ANI     00000100B
0070 C27600             JNZ     $+6         ;SKIP IF 'DAV'/ON
0073 320920             STA     SDOBF
                        ;
```

```
0076 7C                 MOV     A,H
0077 A7                 ANA     A
0078 C28000             JNZ     EDGE
007B 7D                 MOV     A,L
007C A7                 ANA     A
007D CA8401             JZ      SEND      ;SKIP IF NO EDGE
                        ;
                EDGE:   ;
0080 7D                 MOV     A,L       ;RECALL ON EDGE WITHOUT SELF ON
0081 E620               ANI     00100000B
0083 C24301             JNZ     ONIFC
0086 3E01               MVI     A,1
0088 320A20             STA     SNDBF
                        ;
008B 7D                 MOV     A,L
008C E680               ANI     10000000B
008E C2BE00             JNZ     ONATN
                        ;
0091 7D                 MOV     A,L
0092 E604               ANI     00000100B
0094 C2DB00             JNZ     ONDAV
                        ;
0097 7C                 MOV     A,H       ;RECALL OFF EDGE
0098 E680               ANI     10000000B
009A C2EC00             JNZ     OFATN
                        ;
009D 7C                 MOV     A,H
009E E604               ANI     00000100B
00A0 C2EC00             JNZ     OFDAV
                        ;
00A3 7C                 MOV     A,H
00A4 E602               ANI     00000010B
00A6 C22401             JNZ     OFNDAC
                        ;
00A9 7D                 MOV     A,L
00AA E602               ANI     00000010B
00AC C20601             JNZ     ONNDAC
                        ;
00AF 7D                 MOV     A,L
00B0 E601               ANI     00000001B
00B2 C21D01             JNZ     ONNRFD
                        ;
00B5 7C                 MOV     A,H
00B6 E601               ANI     00000001B
00B8 C21D01             JNZ     OFNRFD
                        ;
00BB C38401             JMP     SEND
                ;
                ;
                ONATN:  ;
00BE 3A0320             LDA     OCOBF
00C1 E640               ANI     01000000B     ;ATN,IFC,REN,EOI,DAV = OFF
00C3 F603               ORI     00000011B     ;NDAC,NRFD = ON
00C5 320088             STA     OCB
```

```
00C8 E6FE               ANI     11111110B ;NRFD = OFF
00CA 320320             STA     OCOBF
00CD 320088             STA     OCB
                        ;
00D0 3A0820             LDA     SCOBF
00D3 E6D0               ANI     11010000B ;IFC,EOI,DAV,NDAC,NRFD = OFF
00D5 320820             STA     SCOBF
                        ;
00D8 C38401             JMP     SEND
                ;
                ;
                ONDAV:  ;
00DB 3A0080             LDA     ODB
00DE 320920             STA     SDOBF
                        ;
00E1 3A0820             LDA     SCOBF
00E4 E6DC               ANI     11011100B ;IFC,NDAC,NRFD = OFF
00E6 320820             STA     SCOBF
                        ;
00E9 C38401             JMP     SEND
                ;
                ;
                OFATN:  ;
                OFDAV:  ;
00EC 3A0320             LDA     OCOBF
00EF E6FE               ANI     11111110B ;NRFD = OFF
00F1 F602               ORI     00000010B ;NDAC = ON
00F3 320320             STA     OCOBF
00F6 320088             STA     OCB
                        ;
00F9 3A0820             LDA     SCOBF
00FC E6FC               ANI     11111100B ;NDAC,NRFD = OFF
00FE F602               ORI     00000010B ;NDAC = ON
0100 320820             STA     SCOBF
                        ;
0103 C38401             JMP     SEND
                        ;
                ONNDAC: ;
0106 3A0220             LDA     OCIBF
0109 E601               ANI     00000001B
010B CA8401             JZ      SEND
                        ;
010E AF                 XRA     A
010F 320A20             STA     SNDBF
                        ;
0112 3A0220             LDA     OCIBF
0115 E6FD               ANI     11111101B
0117 320220             STA     OCIBF
                        ;
011A C38401             JMP     SEND
                        ;
                ONNRFD: ;
                OFNRFD: ;
011D AF                 XRA     A
```

〈リスト1〉コントロール・プログラム(つづき)

```
011E 320A20         STA     SNDBF
                    ;
0121 C38401         JMP     SEND
                    ;
            OFNDAC: ;
0124 3A0820         LDA     SCOBF
0127 F601           ORI     00000001B
0129 320820         STA     SCOBF
012C 3A0220         LDA     OCIBF
012F E601           ANI     00000001B
0131 C28401         JNZ     SEND
                    ;
0134 AF             XRA     A
0135 320A20         STA     SNDBF
                    ;
0138 3A0220         LDA     OCIBF
013B F602           ORI     00000010B
013D 320220         STA     OCIBF
                    ;
0140 C38401         JMP     SEND
            ;
            ;
            ONIFC:  ;
0143 AF             XRA     A
0144 320220         STA     OCIBF
0147 320320         STA     OCOBF
014A 320520         STA     LOCIBF
014D 320720         STA     SCIBF
0150 320820         STA     SCOBF
                    ;
0153 320080         STA     ODB
0156 320088         STA     OCB
                    ;
0159 3A00E0         LDA     IFC ;CLEAR IFC LATCH
                    ;
015C 3A0098         LDA     SIOC
015F E601           ANI     1
0161 CA5C01         JZ      $-5
                    ;
0164 3E20           MVI     A,00100000B ;SET IFC
0166 320090         STA     SIOD
                    ;
0169 3A0098         LDA     SIOC
016C E601           ANI     1
016E CA6901         JZ      $-5
                    ;
0171 AF             XRA     A
0172 320090         STA     SIOD
                    ;
0175 3EC8           MVI     A,200
0177 3D             DCR     A
0178 C27701         JNZ     $-1
                    ;
017B 3A0098         LDA     SIOC
017E 3A0090         LDA     SIOD
                    ;
0181 C33300         JMP     MAIN
                    ;
            SEND:   ;
0184 3A0020         LDA     TSTMR
0187 FEFF           CPI     0FFH
0189 CA8D01         JZ      $+4
018C 3C             INR     A
018D 320020         STA     TSTMR
0190 FE15           CPI     21
0192 DABF01         JC      RSV
                    ;
0195 3A0098         LDA     SIOC
0198 E601           ANI     1
019A CABF01         JZ      RSV
                    ;
019D 3A0A20         LDA     SNDBF
01A0 A7             ANA     A
01A1 CABF01         JZ      RSV
                    ;
01A4 3A0820         LDA     SCOBF
01A7 320090         STA     SIOD
                    ;
01AA 3A0098         LDA     SIOC
01AD E601           ANI     1
01AF CAAA01         JZ      $-5
                    ;
01B2 3A0920         LDA     SDOBF
01B5 320090         STA     SIOD
                    ;
01B8 AF             XRA     A
01B9 320020         STA     TSTMR
01BC 320A20         STA     SNDBF
                    ;
                    ;
            RSV:    ;
01BF 3A0120         LDA     RCTMR
01C2 FEFF           CPI     0FFH
01C4 CAC801         JZ      $+4
01C7 3C             INR     A
01C8 320120         STA     RCTMR
01CB FE04           CPI     4
01CD DAE201         JC      RSV2
                    ;
            RSV1:   ;
01D0 3A0098         LDA     SIOC
01D3 E602           ANI     2
01D5 CA3300         JZ      MAIN
                    ;
01D8 3A0090         LDA     SIOD
01DB 320720         STA     SCIBF
                    ;
01DE AF             XRA     A
01DF 320120         STA     RCTMR
                    ;
            RSV2:   ;
01E2 3A0098         LDA     SIOC
01E5 E602           ANI     2
01E7 C2ED01         JNZ     RSV24
01EA C33300         JMP     MAIN
                    ;
            RSV24:  ;
01ED 3A0090         LDA     SIOD
01F0 320620         STA     SDIBF
                    ;
            SEDGE:  ;
01F3 3A0720         LDA     SCIBF
01F6 4F             MOV     C,A
01F7 3A0320         LDA     OCOBF
01FA 47             MOV     B,A
                    ;
01FB A9             XRA     C
01FC 5F             MOV     E,A
                    ;
01FD A1             ANA     C
01FE 6F             MOV     L,A ;SAVE ON EDGE
                    ;
01FF 7B             MOV     A,E
0200 A0             ANA     B
0201 67             MOV     H,A ;SAVE OFF EDGE
                    ;
0202 E6DC           ANI     11011100B
0204 2F             CMA
0205 57             MOV     D,A
0206 3A0220         LDA     OCIBF
0209 A2             ANA     D
020A 320220         STA     OCIBF
                    ;
020D 7D             MOV     A,L
020E E620           ANI     00100000B
0210 C2D702         JNZ     RONIFC
                    ;
0213 7D             MOV     A,L
0214 E680           ANI     10000000B
0216 C24002         JNZ     RONATN
                    ;
0219 7D             MOV     A,L
021A E604           ANI     00000100B
021C C25C02         JNZ     RONDAV
                    ;
021F 7C             MOV     A,H
0220 E680           ANI     10000000B
0222 C2BE02         JNZ     ROFATN
                    ;
0225 7C             MOV     A,H
0226 E604           ANI     00000100B
```

〈リスト1〉コントロール・プログラム(つづき)

```
0228 C2BE02         JNZ   ROFDAV
                    ;
         SOUT:      ;
022B 3A0720         LDA   SCIBF
022E 320320         STA   OCOBF
0231 320088         STA   OCB
                    ;
0234 3A0620         LDA   SDIBF
0237 320420         STA   ODOBF
023A 320080         STA   ODB
                    ;
023D C33300         JMP   MAIN
                    ;
         RONATN:    ;
0240 C32B02         JMP   SOUT
                    ;
0243 3A0720         LDA   SCIBF
0246 320088         STA   OCB
0249 3A0820         LDA   SCOBF
024C E640           ANI   01000000B
024E F602           ORI   00000010B
0250 320820         STA   SCOBF
0253 3A0088         LDA   OCB
0256 320220         STA   OCIBF
                    ;
0259 C32B02         JMP   SOUT
                    ;
         RONDAV:    ;
025C 3A0320         LDA   OCOBF
025F E6FC           ANI   11111100B
0261 320320         STA   OCOBF
0264 57             MOV   D,A
0265 320088         STA   OCB
                    ;
         RONDAV2:   ;
0268 3A0088         LDA   OCB
026B 67             MOV   H,A
026C E603           ANI   00000011B
026E CAA902         JZ    RONDAV6
0271 7C             MOV   A,H
0272 E601           ANI   00000001B
0274 CA8C02         JZ    RONDAV4
                    ;
0277 3A0320         LDA   OCOBF
027A 2F             CMA
027B A4             ANA   H
027C E6A0           ANI   10100000B
027E C23300         JNZ   MAIN
0281 3A0098         LDA   SIOC
0284 E602           ANI   2
0286 C2D001         JNZ   RSV1
0289 C36802         JMP   RONDAV2
                    ;
         RONDAV4:   ;
028C 3A0620         LDA   SDIBF
028F 320420         STA   ODOBF
0292 320080         STA   ODB
                    ;
0295 3A0720         LDA   SCIBF
0298 5F             MOV   E,A
0299 E608           ANI   00001000B
029B B2             ORA   D
029C 320088         STA   OCB
029F 7B             MOV   A,E
02A0 320320         STA   OCOBF
02A3 320088         STA   OCB
                    ;
02A6 C33300         JMP   MAIN
                    ;
         RONDAV6:   ;
02A9 3A0088         LDA   OCB
02AC 320220         STA   OCIBF
                    ;
02AF AF             XRA   A
02B0 320820         STA   SCOBF
02B3 320920         STA   SDOBF
02B6 3E01           MVI   A,1
02B8 320A20         STA   SNDBF
02BB C38401         JMP   SEND
                    ;
         ROFATN:    ;
         ROFDAV:    ;
02BE 3A0320         LDA   OCOBF
02C1 57             MOV   D,A
02C2 3A0720         LDA   SCIBF
02C5 5F             MOV   E,A
02C6 E604           ANI   00000100B
02C8 B2             ORA   D
02C9 57             MOV   D,A
02CA 320088         STA   OCB
                    ;
02CD 7B             MOV   A,E
02CE E608           ANI   00001000B
02D0 B2             ORA   D
02D1 320088         STA   OCB
                    ;
02D4 C32B02         JMP   SOUT
                    ;
         RONIFC:    ;
02D7 AF             XRA   A
02D8 320220         STA   OCIBF
02DB 320320         STA   OCOBF
02DE 320520         STA   LOCIBF
02E1 320820         STA   SCOBF
02E4 320720         STA   SCIBF
                    ;
02E7 3200E0         STA   IFC
02EA 3A00E0         LDA   IFC   ;IFC PULSE
                    ;
02ED AF             XRA   A
02EE 320080         STA   ODB
02F1 320088         STA   OCB
                    ;
02F4 C33300         JMP   MAIN
                    ;
                    ;
02F7                END
```

```
099A CLK      C000 DIPSW    0080 EDGE     E000 IFC      0000 INIT
A000 INTP     2005 LOCIBF   0033 MAIN     8800 OCB      2002 OCIBF
2003 OCOBF    8000 ODB      2004 ODOBF    00EC OFATN    00EC OFDAV
0124 OFNDAC   011D OFNRFD   00BE ONATN    00DB ONDAV    0143 ONIFC
0106 ONNDAC   011D ONNRFD   2000 RAMTOP   2001 RCTMR    02BE ROFATN
02BE ROFDAV   0000 ROMTOP   0240 RONATN   025C RONDAV   0268 RONDAV2
028C RONDAV4  02A9 RONDAV6  02D7 RONIFC   01BF RSV      01D0 RSV1
01E2 RSV2     01ED RSV24    200B SAREA    2007 SCIBF    2008 SCOBF
2006 SDIBF    2009 SDOBF    01F3 SEDGE    0184 SEND     9800 SIOC
9000 SIOD     200A SNDBF    022B SOUT     204B STACK    2000 TSTMR
```

§3-4

HP9816-PC9801間をGPIBを用いてファイル転送する

巨 慎一

現在，パソコンは異機種間の互換性がほとんどなく，ディスク・サイズも3.5インチ，5インチ，8インチとまちまちです．しかし，プログラムやデータ，あるいは，ディスク装置を共用したいことがしばしばあります．

このようなとき，RS-232CやGPIBを使ったパソコン間の通信が便利です．RS-232Cを使ったものについては比較的多く紹介されているので，GPIBによるデータ転送について紹介します(図1)．

接続例として，HP製のHP9816と日本電気製のPC9801を取り上げることにします．

ハードの設定

GPIBは，本来，計測器の制御用バスなので，パソコン同士をつなぐ場合，問題点がないわけではありません．GPIBでは，接続された機器の中でシステム・コントローラは1台でなければなりません．しかし，一般的に，パソコンは，コントローラになっているので，パソコン同士をつなぐ場合，GPIB上に複数のコントローラが存在することになってしまいます．このような場合，GPIB基板上のDIP-SWを使って，一方をスレーブに指定しなければなりません．

ここでは，HP9816をシステム・コントローラ，PC9801をスレーブ・モードとして接続することにします．

◆ PC9801のGPIBカードの設定方法

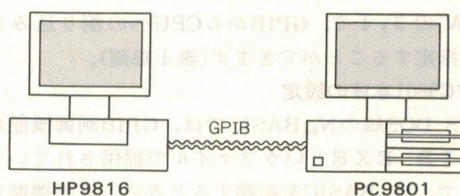
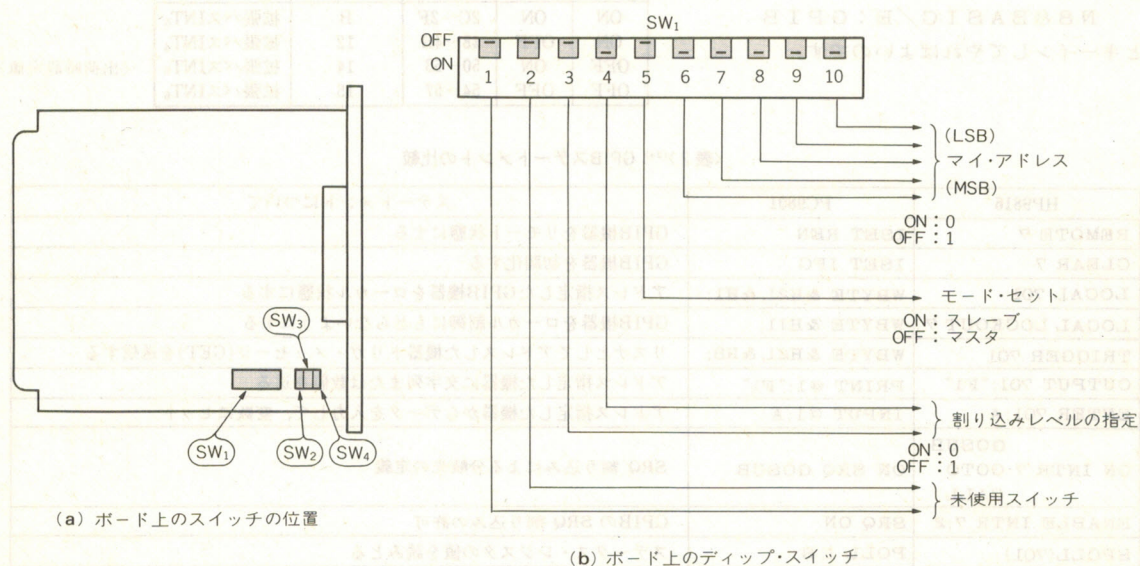
GPIBカードを使うには，カード上のディップ・スイッチ，ジャンパ・スイッチ(図2)とPC9801本体のメモリ・スイッチをプリセットする必要があります．

〈図1〉 GPIBによる接続

〈図2〉[9] PC9801のGPIBカードの設定

SW₁
OFF
ON
1 2 3 4 5 6 7 8 9 10
(LSB)
マイ・アドレス
(MSB)
ON：0
OFF：1
モード・セット
ON：スレーブ
OFF：マスタ
割り込みレベルの指定
ON：0
OFF：1
未使用スイッチ
SW₃
SW₁ SW₂ SW₄

(a) ボード上のスイッチの位置

(b) ボード上のディップ・スイッチ

① マイ・アドレスの設定

SW_1の6～10でPC9801のアドレスを設定します.

② モード・セット

PC9801のGPIBカードには，マスタ・モードとスレーブ・モードと呼ばれる二つのモードがあります.

マスタ・モードというのは，PC9801がシステム・コントローラとなるモードで，各機器をトーカやリスナに指定したりマルチ・ライン・メッセージを送出することができます.

スレーブ・モードというのは，GPIB上のコントローラからの指定により，トーカやリスナになるモードで，PC9801はコントローラにはなりません.

これらのモードは，システムを起動させたときに，セットされていたSW_{1-5}の状態によって決まるので，それ以降はプログラムでモードを変更することはできません.

PC9801を使って測定器をコントロールする場合は，マスタ・モード(SW_{1-5}をOFF)にします．一方，GPIB上にほかのコントローラが存在し，その下でPC9801を働かせるときは，スレーブ・モード(SW_{1-5}をON)にします．HP9816とのファイル転送では，HP9816がコントローラになっているので，PC9801をスレーブ・モードにして使います.

③ 割り込みレベルの設定

SW_1の3，4で，GPIBからCPUへの割り込みレベルを指定することができます(表1参照).

④ PC9801本体の設定

MS-DOS版のN_{88}BASICでは，GPIB制御機能は，ＧＰＩＢ．ＥＸＥというファイルで提供されています．そこで，N_{88}BASICを起動するときに，拡張機能を使うことを宣言しなければなりません.

起動時に，

Ｎ８８ＢＡＳＩＣ／Ｅ：ＧＰＩＢ

とキーインしてやればよいのです.

(注)MS-DOS版でないN_{88}BASICの場合は，PC9801のメモリSW_{4-5}をONにする必要があります.

これで，GPIBカードの設定はOKです.

◆ PC9801のGPIBコマンドについて

HP9816にあるステートメントでPC9801にはないものがありますが，表2に示したようにして対応させることができます.

GPIB制御の測定器では，多くの場合，HP9816によるプログラミング例が付いていますが，この表を参考にすれば，HP9816用のプログラムをPC9801用に書き換えることができます.

◆ PC9801のGPIBプログラム例(スペアナをコントロール)

PC9801を使ったGPIBによる測定器のコントロールと，データ読み取りについて，HP社のスペクトル・アナライザHP8590Aを取り上げて説明します(図3).

「スペアナをプリセットした後でシングル・スイープ・モードに設定する．次にスペアナの中心周波数を300MHzにセットし，1回スイープを行わせる．そして，スペアナの中心周波数を読み取り，画面に表示する.」

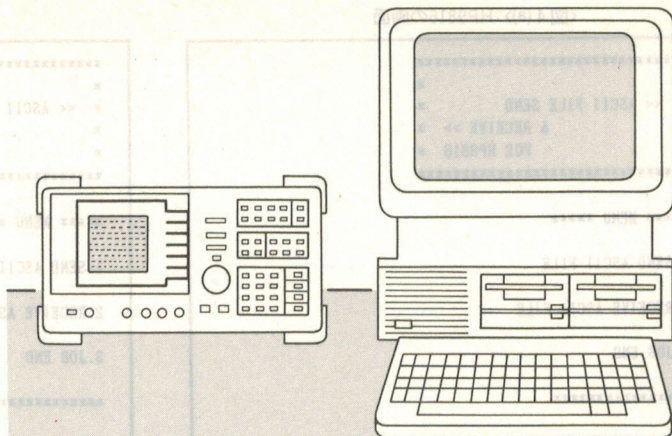
これを行わせるためのHP9816とPC9801のプログラムが，リスト1とリスト2です．HP9816では，ＣＬＥＡＲによってプリセットしますが，PC9801では，ＩＳＥＴ ＩＦＣを使います.

〈表1〉[19] 割り込みレベルの設定

SW_1 3	SW_1 4	アドレス	ベクタ番号	用　途	
ON	ON	2C～2F	B	拡張バスINT_0	
ON	OFF	48～4B	12	拡張バスINT_4	
OFF	ON	50～53	14	拡張バスINT_5	(出荷時設定値)
OFF	OFF	54～57	15	拡張バスINT_6	

〈表2〉[24] GPIBステートメントの比較

HP9816	PC9801	ステートメントについて
REMOTE 7	ISET REN	GPIB機器をリモート状態にする
CLEAR 7	ISET IFC	GPIB機器を初期化する
LOCAL 701	WBYTE &H21, &H1;	アドレス指定したGPIB機器をローカル状態にする
LOCAL LOCKOUT 7	WBYTE &H11;	GPIB機器をローカル制御にもどらないようにする
TRIGGER 701	WBYTE &H21, &H8;	リスナとしてアドレスした機器トリガ・メッセージ(GET)を送信する
OUTPUT 701;"F1"	PRINT @1;"F1"	アドレス指定した機器に文字列または数値を送る
ENTER 701;A	INPUT @1;A	アドレス指定した機器からデータを入力して，変数にセット
ON INTR 7 GOSUB / GOTO / CALL	ON SRQ GOSUB	SRQ 割り込みによる分岐先の定義
ENABLE INTR 7;2	SRQ ON	GPIBの SRQ 割り込みの許可
SPOLL(701)	POLL 1, S	ステータス・レジスタの値を読みとる

〈図3〉
GPIBでスペアナHP8590Aをコントロール

次に，スペアナにコマンド(IP：初期化，SNGLS：シングル・スイープ・モード，CF300MHz：中心周波数を300MHzに設定，TS：スイープ，CF？：中心周波数を出力)を送ります．最後に，スペアナから中心周波数を読み出します．

転送プログラム

HP9816とPC9801をつないで，アスキ・セーブされたプログラムのファイルの転送と，データ・ファイルの転送を行ってみることにします．このとき，HP9816はシステム・コントローラ，PC9801は，スレーブ・モードに設定します．

◈ アスキ・セーブされたファイルの転送

転送プログラム(章末に掲載リスト3，リスト4)を実行すると，画面が図4のようになり送信か受信かを聞いてくるので，1台を送信に，ほかの1台を受信に設定します．次に，転送するファイル名を入力すると転送を開始します．

アスキ・セーブのしかたですが，HP9816では，SAVE“ファイル名”とすると，プログラムをアスキ・セーブすることができます．また，PC9801では，SAVE“ファイル名”，Aとすることによってアスキ・セーブすることができます．

MS-DOS版のN_{88}BASICでは，LOAD命令を実行したとき“.BAS”という拡張子が自動的に付加されてファイル・アクセスが行われます．そこで，HP9816からPC9801にファイルを転送する場合，PC9801側のファイル名に“.BAS”という拡張子を付加しておくとよいでしょう．

◈ データ・ファイルの転送

例として，図5のようにセーブされたデータ・ファイルの転送を行ってみます．

先ほどと同じように 送信，受信の別とファイル名を入力することにより，転送を行うことができます(図6)．セーブされたデータの様式によって，プログラ

〈リスト1〉
HP9816による
プログラム

```
10 SPANA=718                            ’スペアナのＧＰＩＢ上のアドレス 18
20 CLEAR SPANA                          ’スペアナの初期化
30 OUTPUT SPANA;"IP;SNGLS;"             ’初期化、シングルスイープモード
40 OUTPUT SPANA;"CF 300MHz;TS;"         ’中心周波数を300MHzに設定、スイープ
50 OUTPUT SPANA;"CF?;"                  ’中心周波数出力
60 ENTER SPANA;CFRQ                     ’中心周波数の読み出し
70 PRINT "CENTER FREQUENCY =";CFRQ;"Hz"  ’読み出した周波数を画面に出力
80 END
```

〈リスト2〉
PC9801による
プログラム

```
10 SPANA=18                             ’スペアナのＧＰＩＢ上のアドレス 18
20 ISET IFC                             ’スペアナの初期化
30 PRINT @SPANA;"IP;SNGLS;"             ’初期化、シングルスイープモード
40 PRINT @SPANA;"CF 300MHz;TS;"         ’中心周波数を300MHzに設定、スイープ
50 PRINT @SPANA;"CF?;"                  ’中心周波数出力
60 INPUT @SPANA;CFRQ                    ’中心周波数の読み出し
70 PRINT "CENTER FREQUENCY =";CFRQ;"Hz"  ’読み出した周波数を画面に出力
80 END
```

〈図4(a)〉 HP9816の画面

```
********************************
*                              *
*  << ASCII FILE SEND          *
*               & RECEIVE >>   *
*                 FOR HP9816   *
********************************

***** MENU *****

1.SEND ASCII FILE

2.RECEIVE ASCII FILE

3.JOB END

****************

PLEASE INPUT COMMAND =
```

〈図4(b)〉 PC9801の画面

ム(章末に掲載リスト5,リスト6)を適当に変更してやれば，必要なデータの転送が可能となります．

(注)MS-DOS上のN_{88}BASICを使っているので，PC9801のディスク・ドライブの指定は，A:，B:，…としています．

● おわりに

ここでは，GPIBによるデータ転送の基礎的なプログラムを紹介しました．ここで紹介したPC9801用のプログラムはN_{88}BASICで書いてあるので，PC9801同士あるいはPC9801とPC8801のファイル転送やデータ転送にも使うことができます．

この場合も一方をマスタ・モード，他方をスレーブ・モードにしてやる必要があります．また，MS-DOS版のN_{88}BASICかどうかにより，ディスク・ドライブの指定の仕方を変更する必要が生じてきます．

入力したファイルがなかったときやディスクが一杯になってしまったときなどのエラー処理は，ここで紹介したプログラムでは行っていませんが，必要なら自分でトライしてみてください．

また，転送するデータのフォーマットが違う場合は，それに合わせてプログラムを変更してください．

〈図5〉 セーブされるデータ・フォーマット

HEAD(1)	HEAD(2)	…………	HEAD(22)
DATA(1,1)	DATA(1,2)	…………	DATA(1,12)
DATA(2,1)	DATA(2,2)	…………	DATA(2,12)
⋮			
DATA(N,1)	DATA(N,2)	…………	DATA(N,12)

◆参考・引用*文献◆

(1)*松井雅行，竹尾佳己；GPIBコントローラ，トランジスタ技術，1983年1月号．

(2)*松崎幹夫；GPIBの使い方，トランジスタ技術，1983年12月号．

(3)*松井雅行，竹尾佳己；GPIBインターフェース成功への手法，トランジスタ技術，1984年1月号．

(4)*日本電気，μPD7210アプリケーション・ノート，1982年4月号．

(5)*川村道生，檜山哲夫；GPIBインターフェースの製作，トランジスタ技術，1982年1月号．

(6)*Motorola，MC68488データ・シート，1980年．

(7)*TI社，The Bipolar Digital, Integrated Circuits Data Book，1985年．

(8) 鬼頭史郎；標準インターフェース・バス(HP-IB)，インターフェース，1977年，2月号，p.41，CQ出版社．

(9) 岡村廸夫；IEEE-488標準ディジタル・バスとその応用，インターフェース，1979年．2月号，p.45，CQ出版社．

(10) 岡村廸夫；IEEE-488標準ディジタル・バスとその応用，インターフェース，1980年，7月号，p.70，CQ出版社．

(11) 神谷峰夫；GP-IBインターフェースの設計と問題点，インターフェース，1980年，8月号，p.70，CQ出版社．

(12) 岡村廸夫；標準ディジタル・バス(IEEE-488)とそのソフトウェア，インターフェース，1983年，11月号，p.189，CQ出版社．

(13) 佐倉成之；光ファイバ信号伝送回路の製作と実験，トランジスタ技術，1983年，9月号，p.355．

(14) インターシル，IM6402，インターシル・デジタル・プロダクツ，1978年，pp.8～125．

(15)*東芝，東芝光伝送リンク，テクニカル・データ，1982年．

(16) マイクロサイエンス，UIO-488B，取扱説明書．

(17) マイクロサイエンス，SBD-488T，取扱説明書．

〈図6(a)〉 HP9816の画面

```
*******************************
*                             *
*  << DATA FILE SEND          *
*             & RECEIVE >>    *
*               FOR HP9816    *
*******************************

***** MENU *****

1.SEND DATA

2.RECEIVE DATA

3.JOB END

****************

PLEASE INPUT COMMAND =
```

〈図6(b)〉 PC9801の画面

```
*******************************
*                             *
*  << DATA FILE SEND          *
*             & RECEIVE >>    *
*               FOR PC-9801   *
*******************************

***** MENU *****

1.SEND DATA

2.RECEIVE DATA

3.JOB END

****************

PLEASE INPUT COMMAND =
```


(18) シャープ，MZ80B，オーナーズマニュアル/GPIBインターフェース．

(19)*日本電気，GP-IB(IEEE-488)インターフェースボードユーザーズマニュアル，pp.17〜19．

(20) YHP，モデル16 BASIC操作法入門．

(21) YHP，BASIC I/Oプログラミング．

(22) HP，8590A，スペクトラム・アナライザオペレーティング・マニュアル．

(23) HP，8590A Portalbe RF Spectrum Analyzer Programming Manual HP-IB．

(24)*インターフェース別冊 GPIBプログラミング・ノート，インターフェース，1987年，4月号，CQ出版社．

〈リスト3〉HP9816用のアスキ・ファイル転送プログラム

```
1000 !****************************************
1010 !*                                      *
1020 !*  << SEND & RECEIVE ASCII FILE >>     *
1030 !*                          FOR HP9816  *
1040 !****************************************
1050 !
1060 !***** INIT *****
1070 !
1080 DIM Source_file$[6],Desti_file$[6]
1090 DIM D$[255]
1100 !
1110 !////////////////////////////////////////////////////////
1120 !
1130 Pc_addr=711          !PC-9801 HP-IB ADDRESS
1140 CLEAR Pc_addr        !CLEAR HP-IB
1150 !
1160          PRINT "******************************"
1170          PRINT "*                            *"
1180          PRINT "*  <<ASCII FILE SEND         *"
1190          PRINT "*              & RECEIVE>>   *"
1200          PRINT "*                FOR HP9816  *"
1210          PRINT "******************************"
1220          PRINT
1230          !
1240 Menu:    PRINT "***** MENU *****"
1250          PRINT
1260          PRINT "1.SEND ASCII FILE"
1270          PRINT
1280          PRINT "2.RECEIVE ASCII FILE"
1290          PRINT
1300          PRINT "3.JOB END"
1310          PRINT
1320          PRINT "***************"
1330          PRINT
1340          !
1350          INPUT "PLEASE INPUT COMMAND = ",N
1360          IF N<1 OR N>3 THEN Menu
1370          IF N=1 THEN Fsend
1380          IF N=2 THEN Frec
1390          STOP
1400          !
1410 Fsend:     PRINT "****************************"
1420            PRINT "*                          *"
1430            PRINT "*     SEND ASCII FILE      *"
1440            PRINT "*   HP9816 =====>PC-9801   *"
1450            PRINT "*                          *"
1460            PRINT "****************************"
1470            PRINT
1480            !
1490            INPUT "SOURCE_FILE NAME (HP9816)=",Source_fname$
1500            !
1510            INPUT "DESTINATION_FILE NAME (PC-9801)=",Desti_fname$
1520            IF Desti_fname$="" THEN Desti_fname$=Source_fname$
1530            !
1540            PRINT Source_fname$;"(HP9816) =====> ";Desti_fname$&"(PC-
9801)"
1550            PRINT
1560            !
1570            INPUT "    O.K.=====>Y (Yes) ENTER OR N (No) ENTER",K$
1580            IF K$="Y" THEN St_send
1590            GOTO Fsend
1600            !
1610 St_send:   PRINT "..... NOW BEGIN SEND ASCII FILE ..... "
1620            PRINT
1630 !
1640 !********** OPEN_FILE **********
1650 !
1660            ASSIGN @Read_disk TO Source_fname$&":HP82901,700,1"
1670            ON END @Read_disk GOTO F_end
1680            !
1690            OUTPUT Pc_addr;Desti_fname$
1700 Send_loop:ENTER @Read_disk;D$
1710            OUTPUT Pc_addr;D$
1720            GOTO Send_loop
1730            !
1740 F_end:     OUTPUT Pc_addr;"EOF"
1750            ASSIGN @Read_disk TO *
1760            PRINT "..... SEND ASCII FILE END ....."
1770            PRINT
1780            GOTO Menu
1790 !
1800 Frec:      PRINT "****************************"
```

〈リスト3〉HP9816用のアスキ・ファイル転送プログラム(つづき)

```
1810         PRINT "*                         *"
1820         PRINT "*    RECEIVE ASCII FILE    *"
1830         PRINT "*   PC-9801 =====>HP9816   *"
1840         PRINT "*                         *"
1850         PRINT "***************************"
1860         PRINT
1870         !
1880         INPUT "SOURCE_FILE NAME (PC-9801)=",Source_fname$
1890         !
1900         INPUT "DESTINATION_FILE NAME (HP9816)=",Desti_fname$
1910         IF Desti_fname$="" THEN Desti_fname$=Source_fname$
1920         !
1930         PRINT Source_fname$;"(PC-9801) =====> ";Desti_fname$&"(HP
9816)"
1940         PRINT
1950         !
1960         INPUT "    O.K.=====>Y (Yes) ENTER OR N (No) ENTER",K$
1970         IF K$="Y" THEN St_rec
1980         GOTO Frec
1990         !
2000 St_rec:  PRINT "..... NOW BEGIN RECEIVE ASCII FILE ..... "
2010         PRINT
2020 !
2030 !********** OPEN_FILE **********
2040 !
2050         CREATE ASCII Desti_fname$&":HP82901,700,1",144
2060         ASSIGN @Write_disk TO Desti_fname$&":HP82901,700,1"
2070         !
2080         OUTPUT Pc_addr;Source_fname$
2090 Frec_loop:ENTER Pc_addr;D$
2100         IF D$="EOF" THEN Frec_end
2110         OUTPUT @Write_disk;D$
2120         GOTO Frec_loop
2130         !
2140 Frec_end: ASSIGN @Write_disk TO *
2150         PRINT "..... RECEIVE ASCII FILE END ....."
2160         PRINT
2170         GOTO Menu
2180 Imaghead:IMAGE DDDD,"/",DD,"/",DD,"  ",DD,":",DD,":",DD,"  L=",DD,"
  T=",DD,"  W=",DD.DD
2190 END
```

〈リスト4〉PC9801用のアスキ・ファイル転送プログラム

```
1000 '***************************************
1010 '*  << ASCII FILE SEND & RECEIVE >>   *
1020 '*                      FOR PC-9801  *
1030 '***************************************
1040 '
1050 CLS 3
1060 '
1070 PRINT "***************************"
1080 PRINT "*                         *"
1090 PRINT "*  << ASCII FILE SEND      *"
1100 PRINT "*           & RECEIVE >>   *"
1110 PRINT "*              FOR PC-9801 *"
1120 PRINT "***************************"
1130 PRINT
1140 '
1150 *MENU
1160 PRINT "***** MENU *****"
1170 PRINT
1180 PRINT "1.SEND ASCII FILE"
1190 PRINT
1200 PRINT "2.RECEIVE ASCII FILE"
1210 PRINT
1220 PRINT "3.JOB END"
1230 PRINT
1240 PRINT "***************"
1250 PRINT
1260 '
1270 INPUT "PLEASE INPUT COMMAND = ";N
1280 PRINT
1290 IF N<1 OR N>3 THEN *MENU
1300 IF N=3 THEN *JEND
1310 ON N GOSUB *FSEND,*FREC
1320 GOTO *MENU
1330 *JEND
1340 END
1350 '
1360 '***************************
1370 '*                         *
1380 '*    SEND-ASCII FILE      *
1390 '*  PC-9801 ===> HP9816    *
1400 '*                         *
1410 '***************************
```

〈リスト4〉PC9801用のアスキ・ファイル転送プログラム(つづき)

```
1420 '
1430 *FSEND
1440 '
1450 PRINT"***************************"
1460 PRINT"*                         *"
1470 PRINT"*     SEND-ASCII FILE     *"
1480 PRINT"*   PC-9801 ===> HP9816   *"
1490 PRINT"*                         *"
1500 PRINT"***************************"
1510 PRINT
1520 '
1530 INPUT"INSERT FILE TO #2 (PC-9801) O.K. ===> Push ENTER";K$
1540 PRINT
1550 '
1560 LINE INPUT@;PNAME$                          :'ＧＰ－ＩＢよりファイル名入力
1570 PRINT" FILE NAME : ";PNAME$
1580 PRINT
1590 '
1600 '***** OPEN FILE *****
1610 '
1620 OPEN "B:"+PNAME$ FOR INPUT AS #1
1630 '
1640 '*********************
1650 '
1660 PRINT "... NOW SENDING FILE ..."
1670 PRINT
1680 GOSUB *SENDF
1690 '
1700 PRINT "*** FILE SEND COMPLETED ***"
1710 PRINT
1720 '
1730 FOR I=1 TO 1000 :NEXT I
1740 '
1750 CLS 3
1760 RETURN
1770 '
1780 *SENDF
1790 IF EOF(1) THEN *SENDEND
1800 LINE INPUT #1,D$                  :'ファイルより入力
1810 PRINT@;D$                         :'ＧＰ－ＩＢへの出力
1820 GOTO *SENDF
1830 '
1840 *SENDEND
1850 CLOSE
1860 PRINT@;"EOF"                      :'ＧＰ－ＩＢへ" ＥＯＦ"を出力
1870 RETURN
1880 '
1890 '***************************
1900 '*                         *
1910 '*   RECEIVE-ASCII FILE    *
1920 '*   HP9816 ===> PC-9801   *
1930 '*                         *
1940 '***************************
1950 '
1960 *FREC
1970 '
1980 PRINT"***************************"
1990 PRINT"*                         *"
2000 PRINT"*   RECEIVE-ASCII FILE    *"
2010 PRINT"*   HP9816 ===> PC-9801   *"
2020 PRINT"*                         *"
2030 PRINT"***************************"
2040 PRINT
2050 '
2060 INPUT"INSERT FILE TO #2 (PC-9801) O.K. ===> Push ENTER";K$
2070 PRINT
2080 '
2090 LINE INPUT@;PNAME$                          :'ＧＰ－ＩＢよりファイル名入力
2100 PRINT"FILE NAME : ";PNAME$
2110 PRINT
2120 '
2130 '***** OPEN FILE *****
2140 '
2150 OPEN "B:"+PNAME$ FOR OUTPUT AS #1
2160 '
2170 '*********************
2180 '
2190 PRINT "... NOW RECEIVING FILE ..."
2200 PRINT
2210 GOSUB *RECF
2220 '
2230 PRINT "*** FILE RECEIVE COMPLETED ***"
2240 PRINT
2250 '
```

〈リスト4〉PC9801用のアスキ・ファイル転送プログラム(つづき)

```
2260 FOR I=1 TO 1000 :NEXT I
2270 '
2280 CLS 3
2290 RETURN
2300 '
2310 *RECF
2320 LINE INPUT@;D$                      :'GP－IBより入力
2330 IF D$="EOF" THEN *RECEND
2340 PRINT #1,D$                         :'ファイルへの出力
2350 GOTO *RECF
2360 '
2370 *RECEND
2380 CLOSE
2390 RETURN
```

〈リスト5〉HP9816用のデータ転送プログラム

```
1000 !****************************************
1010 !*                                      *
1020 !*   << SEND & RECEIVE DATA FILE >>     *
1030 !*                          FOR HP9816  *
1040 !****************************************
1050 !
1060 !*************** INT. ***************
1070 !
1080         OPTION BASE 1
1090         INTEGER Uti(6),Scanl,Scant
1100         INTEGER Head(22),D(12)
1110         INTEGER Nsize
1120         DIM Source_file$[6],Desti_file$[6]
1130         DIM Dn$[10],Hd$(22)[10],Dat$(1019,12)[10]
1140 !
1150 !/////////////////////////////////////////////////////
```

〈リスト5〉HP9816用のデータ転送プログラム(つづき)

```
1160 !
1170           Pc_addr=711          !PC-9801 HP-IB ADDRESS
1180           CLEAR Pc_addr        !CLEAR HP-IB
1190           !
1200           PRINT "*******************************"
1210           PRINT "*                             *"
1220           PRINT "*  <<DATA FILE SEND           *"
1230           PRINT "*               & RECEIVE>>   *"
1240           PRINT "*                 FOR HP9816  *"
1250           PRINT "*******************************"
1260           PRINT
1270           !
1280 Menu:     PRINT "***** MENU *****"
1290           PRINT
1300           PRINT "1.SEND DATA FILE"
1310           PRINT
1320           PRINT "2.RECEIVE DATA FILE"
1330           PRINT
1340           PRINT "3.JOB END"
1350           PRINT
1360           PRINT "****************"
1370           PRINT
1380           !
1390           INPUT "PLEASE INPUT COMMAND = ",N
1400           IF N<1 OR N>3 THEN Menu
1410           IF N=1 THEN Fsend
1420           IF N=2 THEN Frec
1430           STOP
1440           !
1450 Fsend:    PRINT "****************************"
1460           PRINT "*                          *"
1470           PRINT "*      SEND DATA FILE      *"
1480           PRINT "*   HP9816 =====>PC-9801   *"
1490           PRINT "*                          *"
1500           PRINT "****************************"
1510           PRINT
1520           !
1530           INPUT "SOURCE_FILE NAME (HP9816)=",Source_fname$
1540           !
1550           INPUT "DESTINATION_FILE NAME (PC-9801)=",Desti_fname$
1560           IF Desti_fname$="" THEN Desti_fname$=Source_fname$
1570           !
```

〈リスト5〉HP9816用のデータ転送プログラム(つづき)

```
1580            PRINT Source_fname$;"(HP9816) =====> ";Desti_fname$&"(PC-
9801)"
1590            PRINT
1600            !
1610            INPUT "    O.K.=====>Y (Yes) ENTER OR N (No) ENTER",K$
1620            IF K$="Y" THEN St_send
1630            GOTO Fsend
1640            !
1650 St_send:   PRINT "..... NOW BEGIN SEND DATA FILE ..... "
1660            PRINT
1670 !
1680 !********** OPEN_FILE **********
1690 !
1700            ASSIGN @Read_disk TO Source_fname$&":HP82901,700,1"
1710            ON END @Read_disk GOTO No_data
1720            ENTER @Read_disk;Head(*)
1730            !
1740            FOR I=1 TO 6
1750                Uti(I)=Head(I)
1760            NEXT I
1770            Scanl=Head(7)
1780            Scant=Head(8)
1790            Scanw=Head(9)*.001
1800            PRINT USING Imaghead;Uti(*),Scanl,Scant,Scanw
1810            Nsize=Scanl*Scant-1        !DATA NUMBER=Nsize
1820            PRINT "Nsize=";Nsize
1830            OUTPUT Pc_addr;Desti_fname$
1840            OUTPUT Pc_addr;Nsize
1850            FOR I=1 TO 22
1860               OUTPUT Pc_addr;Head(I)
1870            NEXT I
1880            FOR I=1 TO Nsize
1890                DISP I
1900                IF (I MOD 10)=0 THEN
1910                   PRINT "*";
1920                   IF (I MOD 500)=0 THEN PRINT
1930                END IF
1940                ENTER @Read_disk;D(*)
1950                FOR J=1 TO 12
1960                   OUTPUT Pc_addr;D(J)
1970                NEXT J
1980            NEXT I
1990            !
2000 No_data:   ASSIGN @Read_disk TO *
2010            PRINT
2020            PRINT "..... SEND DATA FILE END ....."
2030            PRINT
2040            GOTO Menu
2050 !
2060 Frec:      PRINT "****************************"
2070            PRINT "*                          *"
2080            PRINT "*     RECEIVE DATA FILE    *"
2090            PRINT "*   PC-9801 =====>HP9816   *"
2100            PRINT "*                          *"
2110            PRINT "****************************"
2120            PRINT
2130            !
2140            INPUT "SOURCE_FILE NAME (PC-9801)=",Source_fname$
2150            !
2160            INPUT "DESTINATION_FILE NAME (HP9816)=",Desti_fname$
2170            IF Desti_fname$="" THEN Desti_fname$=Source_fname$
2180            !
2190            PRINT Source_fname$;"(PC-9801) =====> ";Desti_fname$&"(HP
9816)"
2200            PRINT
2210            !
2220            INPUT "    O.K.=====>Y (Yes) ENTER OR N (No) ENTER",K$
2230            IF K$="Y" THEN St_rec
2240            GOTO Frec
2250            !
2260 St_rec:    PRINT "..... NOW BEGIN RECEIVE DATA FILE ..... "
2270            PRINT
2280 !
2290 !********** OPEN_FILE **********
2300 !
2310            CREATE BDAT Desti_fname$&":HP82901,700,1",144
2320            ASSIGN @Write_disk TO Desti_fname$&":HP82901,700,1"
2330            !
2340            OUTPUT Pc_addr;Source_fname$
2350            !
2360            ENTER Pc_addr;Dn$
2370            Nsize=VAL(Dn$)
2380            PRINT "NSIZE=";Nsize
```

〈リスト5〉HP9816用のデータ転送プログラム(つづき)

```
2390        !
2400        ENTER Pc_addr;Hd$(*)
2410        !
2420        FOR I=1 TO Nsize
2430          DISP I
2440          FOR J=1 TO 12
2450            ENTER Pc_addr;Dat$(I,J)
2460          NEXT J
2470        NEXT I
2480        FOR I=1 TO 22
2490          Head(I)=VAL(Hd$(I))
2500        NEXT I
2510        OUTPUT @Write_disk;Head(*)
2520        !
2530        FOR I=1 TO 6
2540          Uti(I)=Head(I)
2550        NEXT I
2560        Scanl=Head(7)
2570        Scant=Head(8)
2580        Scanw=Head(9)*.001
2590        PRINT USING Imaghead;Uti(*),Scanl,Scant,Scanw
2600        PRINT
2610        !
2620        PRINT "..... NOW WRITE DATA TO DISK ....."
2630        PRINT
2640        !
2650        FOR I=1 TO Nsize
2660            DISP I
2670            FOR J=1 TO 12
2680                D(J)=VAL(Dat$(I,J))
2690            NEXT J
2700            OUTPUT @Write_disk;D(*)
2710        NEXT I
2720        !
2730        ASSIGN @Write_disk TO *
2740        PRINT "..... RECEIVE DATA FILE END ....."
2750        PRINT
2760        GOTO Menu
2770 Imaghead:IMAGE DDDD,"/",DD,"/",DD,"  ",DD,":",DD,":",DD,"  L=",DD,"
  T=",DD,"  W=",DD.DD
2780 END
```

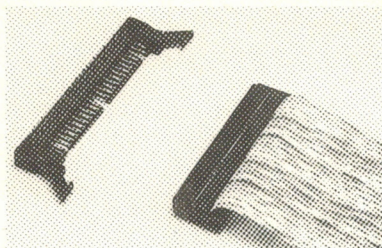
〈リスト6〉PC9801用のデータ転送プログラム

```
1000 '********************************
1010 '*  << DATA SEND & RECEIVE >>   *
1020 '*                FOR PC-9801   *
1030 '********************************
1040 '
1050 CLS 3
1060 DIM HEAD(22),DD(12)
1070 DIM HD$(22),DAT$(12)
1080 '
1090 PRINT "*************************"
1100 PRINT "*                       *"
1110 PRINT "*  << DATA SEND         *"
1120 PRINT "*           & RECEIVE >> *"
1130 PRINT "*             FOR PC-9801  *"
1140 PRINT "*************************"
1150 PRINT
1160 '
1170 *MENU
1180 PRINT "***** MENU *****"
1190 PRINT
1200 PRINT "1.SEND DATA"
1210 PRINT
1220 PRINT "2.RECEIVE DATA"
1230 PRINT
1240 PRINT "3.JOB END"
1250 PRINT
1260 PRINT "****************"
1270 PRINT
1280 '
1290 INPUT "PLEASE INPUT COMMAND = ";N
1300 PRINT
1310 IF N<1 OR N>3 THEN *MENU
1320 IF N=3 THEN *JEND
1330 ON N GOSUB *DSEND,*DREC
1340 GOTO *MENU
1350 *JEND
1360 END
1370 '
1380 '***************************
1390 '*                         *
1400 '*        SEND-DATA        *
```

〈リスト6〉PC9801用のデータ転送プログラム(つづき)

```
1410 '*   PC-9801 ===> HP9816   *
1420 '*                         *
1430 '***************************
1440 '
1450 *DSEND
1460 '
1470 PRINT"***************************"
1480 PRINT"*                         *"
1490 PRINT"*        SEND-DATA        *"
1500 PRINT"*   PC-9801 ===> HP9816   *"
1510 PRINT"*                         *"
1520 PRINT"***************************"
1530 PRINT
1540 '
1550 INPUT"INSERT DATA FILE TO #2 (PC-9801) O.K. ===> Push ENTER";K$
1560 PRINT
1570 '
1580 LINE INPUT@;DNAME$                 :'ＧＰ－ＩＢよりファイル名入力
1590 PRINT"SEND DATA FILE NAME : ";DNAME$
1600 PRINT
1610 '
1620 '***** OPEN FILE *****
1630 '
1640 OPEN "B:"+DNAME$ FOR INPUT AS #1
1650 '
1660 '*********************
1670 '
1680 FOR I=1 TO 22
1690    INPUT #1,HEAD(I)                :'ファイルよりヘッド入力
1700 NEXT I
1710 '
1720 PRINT HEAD(1);"/";HEAD(2);"/";HEAD(3);" ";
1730 PRINT HEAD(4);":";HEAD(5);":";HEAD(6);
1740 PRINT " L=";HEAD(7);" T=";HEAD(8);" W=";HEAD(9)*.001
1750 PRINT
1760 '
1770 DN=HEAD(7)*HEAD(8)-1
1780 '
1790 NSIZE$=STR$(DN)
1800 PRINT@;NSIZE$                      :'ＧＰ－ＩＢへデータ数出力
1810 PRINT"Nsize=";NSIZE$
1820 PRINT
1830 '
1840 FOR I=1 TO 22
1850    HD$(I)=STR$(HEAD(I))
1860    PRINT@;HD$(I)                   :'ＧＰ－ＩＢへヘッド出力
1870 NEXT I
1880 '
1890 FOR I=1 TO DN
1900    IF (I MOD 10)=0 THEN PRINT"*";
1910    IF (I MOD 500)=0 THEN PRINT
1920    FOR J=1 TO 12
1930       INPUT #1,DD(J)               :'ファイルよりデータ入力
1940    NEXT J
1950    FOR J=1 TO 12
1960       DAT$(J)=STR$(DD(J))
1970       PRINT@;DAT$(J)               :'ＧＰ－ＩＢへデータ出力
1980    NEXT J
1990 NEXT I
2000 '
2010 '***** CLOSE FILE ****
2020 '
2030 CLOSE
2040 '
2050 '*********************
2060 '
2070 PRINT
2080 PRINT"DATA SEND END"
2090 PRINT
2100 RETURN
2110 '
2120 '***************************
2130 '*                         *
2140 '*      RECEIVE-DATA       *
2150 '*   HP9816 ===> PC-9801   *
2160 '*                         *
2170 '***************************
2180 '
```

〈リスト６〉PC9801用のデータ転送プログラム(つづき)

```
2190 *DREC
2200 '
2210 PRINT"****************************"
2220 PRINT"*                          *"
2230 PRINT"*      RECEIVE-DATA        *"
2240 PRINT"*   HP9816 ===> PC-9801    *"
2250 PRINT"*                          *"
2260 PRINT"****************************"
2270 PRINT
2280 '
2290 INPUT"INSERT DATA FILE TO #2 (PC-9801) O.K. ===> Push ENTER";K$
2300 PRINT
2310 '
2320 LINE INPUT@;DNAME$                    :'ＧＰ－ＩＢよりファイル名入力
2330 PRINT"RECEIVE DATA FILE NAME : ";DNAME$
2340 PRINT
2350 '
2360 INPUT@;DN                             :'ＧＰ－ＩＢよりデータ数入力
2370 PRINT"Nsize=";DN
2380 PRINT
2390 '
2400 FOR I=1 TO 22
2410     INPUT@;HEAD(I)                    :'ＧＰ－ＩＢよりヘッド入力
2420 NEXT I
2430 '
2440 PRINT HEAD(1);"/";HEAD(2);"/";HEAD(3);" ";
2450 PRINT HEAD(4);":";HEAD(5);":";HEAD(6);
2460 PRINT " L=";HEAD(7);" T=";HEAD(8);" W=";HEAD(9)*.001
2470 PRINT
2480 '
2490 '***** OPEN FILE *****
2500 '
2510 OPEN "B:"+DNAME$ FOR OUTPUT AS #1
2520 '
2530 '*********************
2540 '
2550 FOR I=1 TO 22
2560     WRITE #1,HEAD(I)                  :'ファイルへヘッド出力
2570 NEXT I
2580 '
2590 FOR I=1 TO DN
2600    IF (I MOD 10)=0 THEN PRINT"*";
2610    IF (I MOD 500)=0 THEN PRINT
2620    FOR J=1 TO 12
2630        INPUT@;DD(J)                   :'ＧＰ－ＩＢよりデータ入力
2640    NEXT J
2650    FOR J=1 TO 12
2660        WRITE #1,DD(J)                 :'ファイルへデータ出力
2670    NEXT J
2680 NEXT I
2690 '
2700 '***** CLOSE FILE ****
2710 '
2720 CLOSE
2730 '
2740 '*********************
2750 '
2760 PRINT
2770 PRINT"DATA RECEIVE END"
2780 PRINT
2790 RETURN
```

§4-1

SCSIインターフェースの基礎

里　和政

SCSI(Small Computer System Interface)は，この意味のとおりマイクロコンピュータ・システム向けに考案されたインターフェースです．標準化も進んでおり，1986年7月にANSI(米国規格協会)において規格化されました．

SCSIの基礎は，ハード・ディスク装置用のインターフェースとして使用されていた，SASI(Shugart Associates System Interface)を拡張しています．

SASIは，複雑な周辺装置の制御をホストで行うのではなくコントローラに任せ，簡単なインターフェースによって制御する方法です．

最近では，SCSIを標準とした周辺装置が多くあり，ハード・ディスク装置，光ディスク装置，CD-ROM，プリンタ，計測器などが代表的なものです．

● SCSIバスの構成

SCSIの長所は，SCSIバス上に多くの装置を接続して，同一のプロトコルで制御でき，高速で大量のデータ転送も行えます．同一バス上には，最大8台まで接続することが可能です．

SCSIバス上の装置には，すべてIDと呼ばれる0～7(データ・バスに対応)の装置番号をもっており，この番号によって，装置の識別を行います．

実際の例を図1に示します．この例は，もっとも簡単な構成です．

ホスト側のSCSIの制御を行うコントローラなどをホスト・アダプタと呼びます．

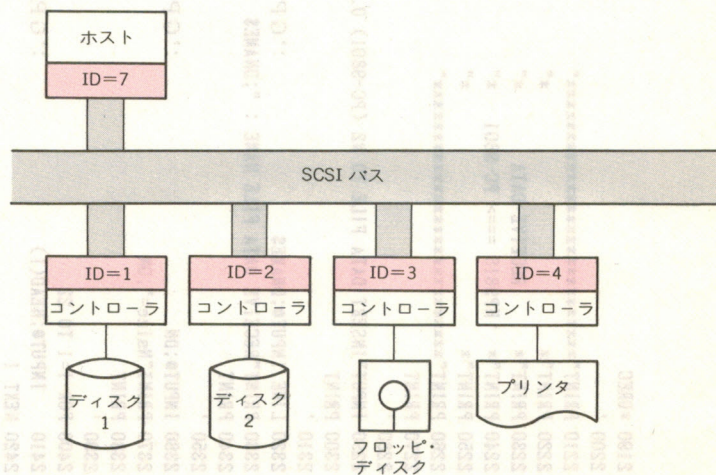
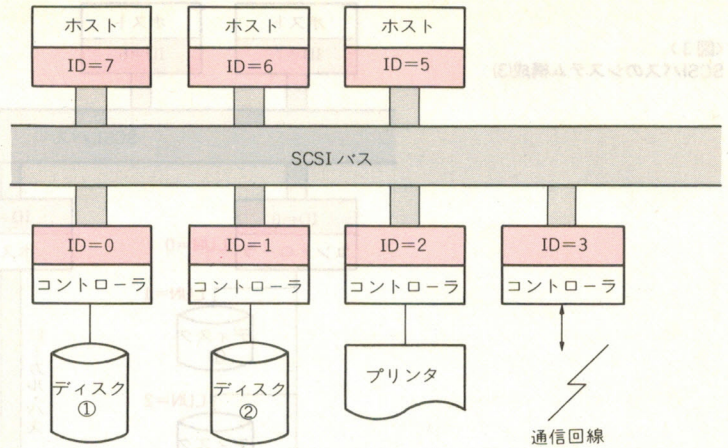
図2は，SCSIの特徴を出した構成です．複数のホスト・コンピュータ，複数の周辺装置を接続することができ，SASIと大きく異なります．

SCSIバス上では，ホストなどの命令を出す側をイニシエータと呼び，ディスク装置などの命令を受け取る側をターゲットと呼びます．しかし，イニシエータであれ，ターゲットであれ装置にはかわりはありません．

ホストは，ある時イニシエータであり，またある時ターゲットである場合が存在します．

装置には，論理ユニット番号(LUN)が付けられており，最大7台までのユニットを接続することができます．LUNを使うことにより図3のような構成も可能です．

SCSIの特徴は，マルチタスク・システムでその能力を発揮することです．なぜならば，以前までのインターフェースでは，ひとつの装置が処理中であれば，その装置がバスを占有して，ほかの装置の処理が行えませんでしたが，SCSIでは，必要に応じてバスを解放することができるからです．

〈図1〉
SCSIバスのシステム構成(1)

<図2>
SCSIバスのシステム構成(2)

SCSIバスの信号

SCSIバスは，9本のデータ線(うち1本は，パリティ線)と9本の制御線で構成されます．

● **BSY信号**：SCSIバスを使用中であることを，ほかのコントローラに示します．またバスを占有する場合に，使用権の調停を行うときに使います．

論理ユニット

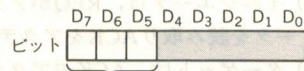
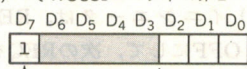
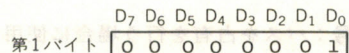
論理ユニットとは，一つのSCSIデバイスがコントロールしている物理的またはバーチャルな装置のことをいいます．通常8台まで(ユニット0～ユニット7)が，SCSIコマンド中図A(a)または"アイデンティファイ"メッセージ中図(b)で，ベンダ・ユニークに指定されます．

普通の小規模システムでは，これでも十分おつりがくるほどですが，SCSI規格では，名前に反して(？)2048台までの論理ユニット番号が，オプションで指定できるようになっています．これは，拡張メッセージである"エクステンデッド・アイデンティファイ(Extended Identify)"メッセージで行います．

4バイト長の拡張メッセージで，第1バイトは拡張メッセージであることを示し(01H)，第2バイトはメッセージの長さ(トータル・バイト数－2＝02H)，第3バイトが"エクステンデッド・アイデンティファイ"のコード(02H)，そして第4バイトで一つの論理ユニット中の〔これは(A)または(B)で指定される〕0～255までのサブ論理ユニット番号を指定します図(c)．これによって，一つのSCSIデバイスは，

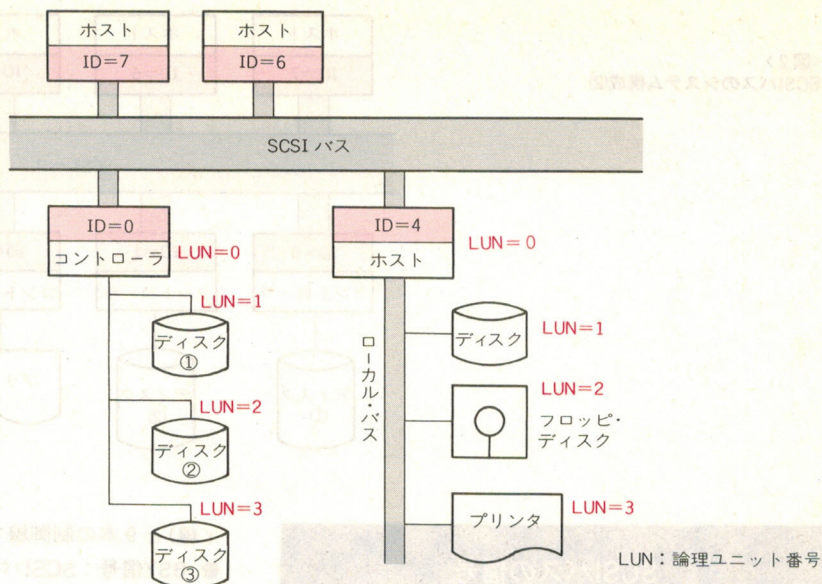
8 × 256 ＝2048
↑論理ユニット ↑サブ論理ユニット

までの論理ユニットを扱うことができます．注意したいのは，SCSIバス上には，あくまで8台までのSCSIデバイス(イニシエータ＋ターゲット)しか接続できないということです．

なお参考までに，GPIB(IEEE-488)では同一バス上には最大15デバイスまで，また，1デバイスには，31台までの2次アドレスで指定できるユニットが接続できます．

(a) すべてのSCSIコマンドの第2バイト

(b) "アイデンティファイ"メッセージ・バイト

	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	
第1バイト	0 0 0 0 0 0 0 1	＝01H
第2バイト	0 0 0 0 0 0 1 0	＝02H
第3バイト	0 0 0 0 0 0 1 0	＝02H
第4バイト		

8ビットのサブ論理ユニット番号(0～255)

(c) "エクステンデッド・アイデンティファイ"メッセージ

〈図3〉
SCSIバスのシステム構成(3)

〈表1〉
MSG, C/D, I/O信号とフェーズ

フェーズ名	MSG	C/D	I/O	転送方向 I　T
コマンド・フェーズ	0	1	0	→
ステータス・フェーズ	0	1	1	←
データ・イン・フェーズ	0	0	1	←
データ・アウト・フェーズ	0	0	0	→
メッセージ・イン・フェーズ	1	0	1	←
メッセージ・アウト・フェーズ	1	0	0	→

I：イニシエータ
T：ターゲット

SCSIバスは負理論であるため，“0”で“H”，“1”で“L”になる．

● **ATN信号**：必要に応じて，いつでもイニシエータから，ターゲットに送ることができます．これは，ターゲットがフェーズの優先権をもっているため，イニシエータからの要求メッセージ(メッセージ・アウト)を送る場合使用します．

● **RST信号**：SCSIバスの解放をおこないます．この信号がアクティブとなるとすべての装置は，その状態にかかわらずバスの解放をしなければなりません．したがって，不用意にこの信号をアクティブにすると，ほかの装置の処理を中止させることになります．

● **MSG, C/D, I/O信号**：フェーズの状態を示します．これらの信号によって実行中のフェーズを示します(表1)．

● **SEL信号**：バスを占有を行う場合に使用します．イニシエータまたはターゲットがバスを使用するとき，セレクション，リセクション・フェーズに入って，バスの確保を行います．

● **REQ, ACK信号**：データ転送のハンドシェイクを制御します．データの転送も同様にターゲット側が権利をもっています．

◆ データ・アウトの手順(イニシエータ・データ送信，図4)

(1) ターゲットがREQをアクティブにする．
(2) イニシエータは，データを出力して，ACKをアクティブにする．
(3) ターゲットは，ACKがアクティブになるとデータを読み取り，REQをOFFにする．
(4) イニシエータは，REQがOFFになるとACKをOFFにして，次のREQを待つ．

◆ データ・インの手順(イニシエータ・データ受信，図5)

(1) ターゲットがデータを出力してREQをアクティブにする．
(2) イニシエータは，REQがアクティブになると，データを読み取りACKをアクティブにする．
(3) ターゲットは，ACKがアクティブになると，REQをOFFにする．
(4) イニシエータは，REQがOFFになるとACKをOFFにして，次のREQを待つ．

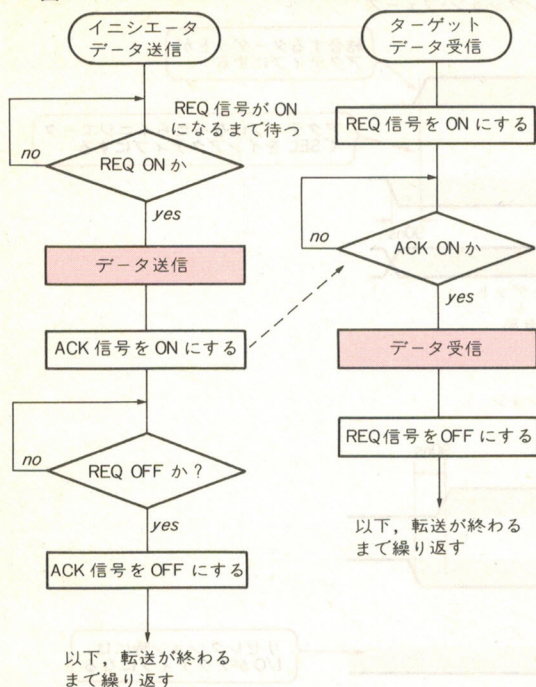
以上の手順にしたがってハンドシェイクを行います．

バスの各フェーズ

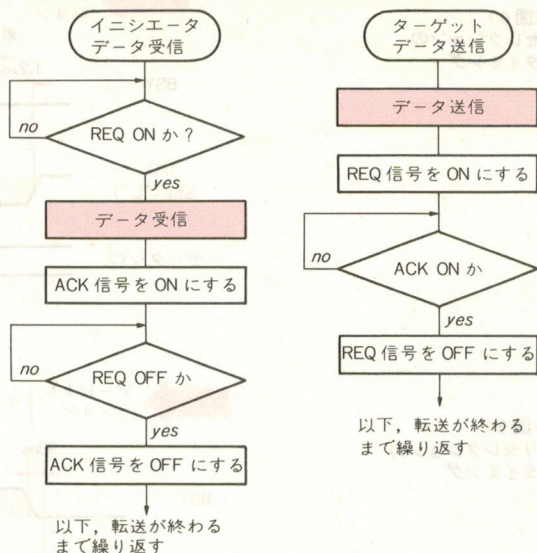
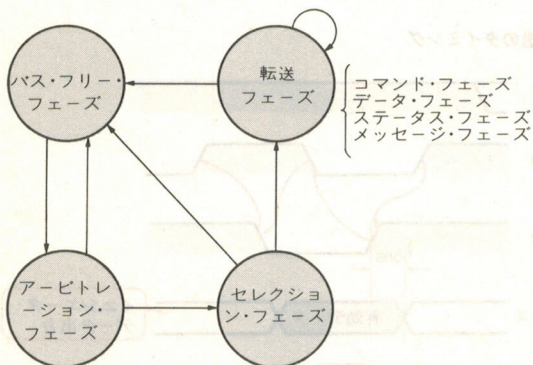
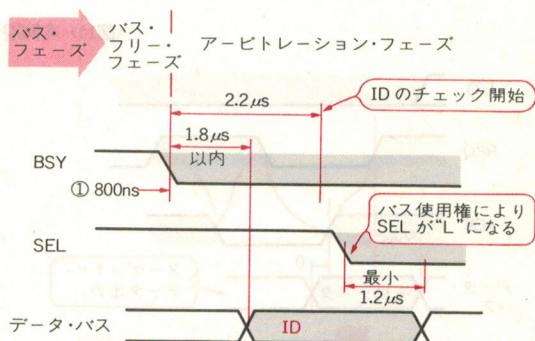
SCSIバスは，イニシエータとターゲット間をフェ

〈図4〉イニシエータのデータ送信手順のフローチャート

イニシエータ
データ送信
REQ信号がON
になるまで待つ
REQ ONか
no
yes
データ送信
ACK信号をONにする
REQ OFFか？
no
yes
ACK信号をOFFにする
以下，転送が終わる
まで繰り返す

ターゲット
データ受信
REQ信号をONにする
ACK ONか
no
yes
データ受信
REQ信号をOFFにする
以下，転送が終わる
まで繰り返す

〈図5〉イニシエータのデータ受信のフローチャート

イニシエータ
データ受信
REQ ONか？
no
yes
データ受信
ACK信号をONにする
REQ OFFか
no
yes
ACK信号をOFFにする
以下，転送が終わる
まで繰り返す

ターゲット
データ送信
データ送信
REQ信号をONにする
ACK ONか
no
yes
REQ信号をOFFにする
以下，転送が終わる
まで繰り返す

〈図6〉フェーズの移行

〈図7〉アービトレーションのタイミング

① バス・フリー検出後，最小800ns(バス・フリー・ディレイ)

ーズによって制御します．

図6にバス・フェーズを示します．

● バス・フリー・フェーズ

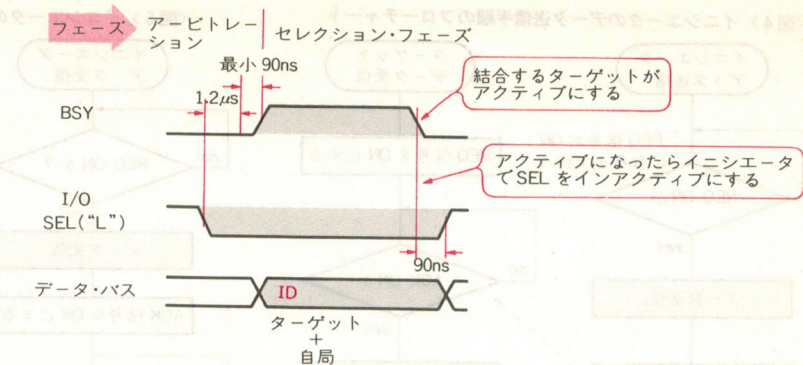
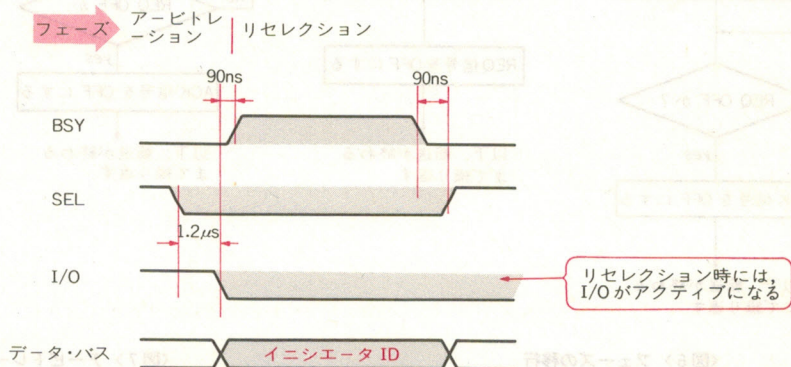
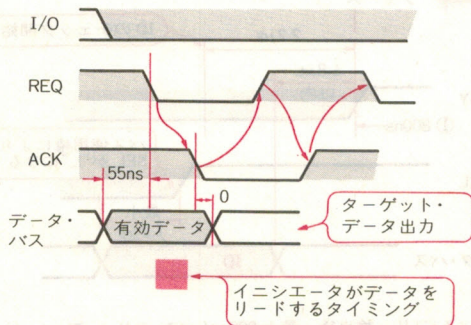
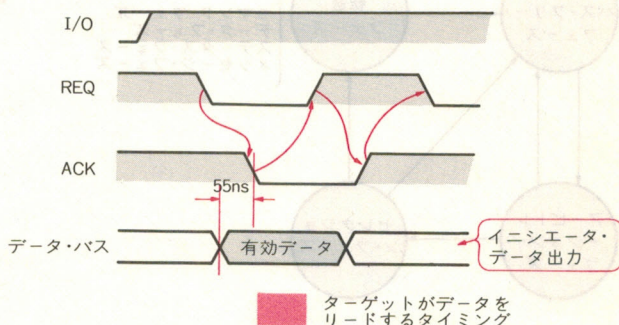
SCSIバスをだれも使用していない状態です．ハードウェア，ソフトウェア・リセット後には，本フェーズに入ります．この状態では，BSYがOFFになっています．

● アービトレーション・フェーズ

本フェーズによってバスの使用権を獲得します．このフェーズは，バス・フリーの状態から入ることができます．

バスを使用したい装置は，BSYをアクティブにして自分の装置番号をデータ・バス上に出力します．このときデータ・バス上のIDは，ビット0〜ビット7のどれかに対応しています．

ほかの装置も同様にバスを使用したいならば，IDを出力します．最初にBSYがアクティブになってから2.2μs後に，アービトレーション(調停)が行われます．アービトレーションでは，自分よりも大きいIDが出力されているかを調べ，もし自分よりも大きいIDがなければ，バスの占有権を得てSELをアクティブにして，次のセレクション・フェーズまたはリセレクション・フェーズを開始します．

このフェーズは，複数のホスト(イニシエータ)があった場合に必要になり，単一のホストではとくに必要

〈図10〉 非同期データ転送のタイミング

（1） ターゲット→イニシエータ　　（2） イニシエータ→ターゲット

なく，次のセレクション・フェーズを開始します(図7)．

● セレクション・フェーズ

アービトレーションによってバスを占有したイニシエータは，自分のIDとアクセスしたい装置のIDをバス上に出力し，BSYをOFFにします．それにより，アクセスされた装置は，結合可能であればBSYをアクティブにして結合します．その後，SELをOFFにします．

自分のIDは，リセレクション・フェーズがあった場合に使用されます(図8)．

以上で，バス上の接続を完了し，コマンド，データの転送を行います．

● リセレクション・フェーズ

セレクションと同様ですが，ターゲット自らバスの占有を行う点が異なります．この機能が，従来のSASIと大きく違う所です．前にも述べたように，デバイス上で時間のかかる処理を行うとバスが不要に占

〈図11〉 コマンド・グループCDBの型式

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	コマンド・コード				
1	LUN			論理ブロック・アドレス(MSB)				
2	論理ブロック・アドレス							
3	論理ブロック・アドレス(LSB)							
4	転送サイズ							
5	コントロール・バイト						フラグ	リンク

(a) グループ0 CDB

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	1	コマンド・コード				
1	LUN			—				REL
2	論理ブロック・アドレス(MSB)							
3	論理ブロック・アドレス							
4	論理ブロック・アドレス							
5	論理ブロック・アドレス(LSB)							
6	—							
7	転送サイズ(MSB)							
8	転送サイズ(LSB)							
9	コントロール・バイト						フラグ	リンク

(b) グループ1 CDB

バイト＼ビット	7	6	5	4	3	2	1	0
0	1	0	1	コマンド・コード				
1	LUN			—				REL
2	論理ブロック・アドレス(MSB)							
3	論理ブロック・アドレス							
4	論理ブロック・アドレス							
5	論理ブロック・アドレス(LSB)							
6	—							
7	—							
8	—							
9	転送サイズ(MSB)							
10	転送サイズ(LSB)							
11	コントロール・バイト						フラグ	リンク

(c) グループ5 CDB

〈表2〉 コマンド・グループCDBのグループ

コマンド・グループ	CDBバイト数
グループ0	6
グループ1	10
グループ2	未定義
グループ3	未定義
グループ4	未定義
グループ5	12
グループ6	定義可能
グループ7	定義可能

〈表3〉 メッセージ・コード

No.	意味
00	コマンド終了
02	セーブ・データ・ポインタ
03	リストア・ポインタ
04	ディスコネクト
06	アボート
07	メッセージ・リジェクト
08	ノー・オペレーション
0A	リンク・コマンド終了
0B	リンク・コマンド終了(フラグ)
0C	バス・デバイス・リセット

有されるため，一度バスを解放して処理が終わった時点でターゲットがバスを占有します．この動作をリセレクションと呼びます．

このときのIDは，結合するイニシエータだけです(図9)．

● コマンド・フェーズ

ターゲットにコマンドを送るフェーズです．セレクション完了後ターゲットは，コマンド待ちになります．コマンドは，CDB(コマンド記述ブロック)と呼ばれるテーブル形式で送ります．またCDBは，コマンド・グループによってCDBのバイト数が異なります．転送タイミングを図10に示します．このタイミングは，各フェーズ同じであり，コントロール信号のMSG, C/D, I/Oが異なるだけです．

表2にコマンド・グループを示します(図11参照)．

● データ・フェーズ

ターゲットは，コマンドを受け取るとデータ・フェーズに移ります．データ・フェーズは，イニシエータとターゲット間でデータの転送を行い，その転送数は，CDB中で指定します．

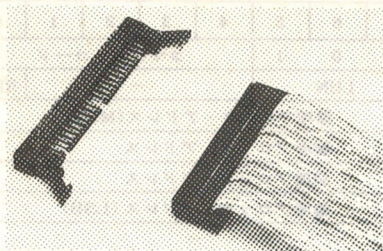
● ステータス・フェーズ

コマンドの終了結果をイニシエータに送るフェーズです．終了結果は，1バイトで示され，正常終了時＝OOHです．

● メッセージ・フェーズ

メッセージ・フェーズには，メッセージ・インとメッセージ・アウトがあります．メッセージ・インは，最後のフェーズであり，インターフェース上のメッセージを1バイト分ターゲットから送ります．SCSIでは，デバイスのアクセスに加えてインターフェース上の管理機能もあるからです．表3にメッセージ・コードを示します．

メッセージ・アウト・フェーズは，イニシエータがターゲットに対して要求メッセージを送る場合に使います．セレクション・フェーズ中にATNをアクティブにすることによってフェーズを終了後，本フェーズに移行します．1バイトのメッセージを送るとコマンド・フェーズに移行します．

以上のフェーズによって，イニシエータとターゲットのデータ伝送を行います．

§4-2 SCSIコントロールLSIの使い方

里 和政/清水哲夫

最近各メーカからSCSI用のコントローラが発表されていますが，国産の物が少ないようです．筆者は，富士通製のSCSIプロトコル・コントローラ（MB89352）を入手して，これを用いてPC9801用のSCSIボードを製作しました（次章を参照）．

現在入手できるコントロールLSIを表1に示します．

MB89352

MB89352は，1987年に発表された新しいLSIです．この前にMB89351がありますが，これは，SCSIバス・ドライバが外部に必要になります．パッケージも大きめの64ピンDIPです．

● MB89352の特徴

MB89352の機能は，MB89351と同様ですが，SCSIバス・ドライバ/レシーバが内蔵されています．そのためパッケージは，48ピンDIPになっています．

このLSIは，同期転送を除いたSCSI規格をLSI上でサポートしており，ソフトウェアでコマンドを与えることによって，容易にアービトレーション，セレクション，リセレクションなどのフェーズの実行，またはイニシエータ，ターゲットなどの動作を行うことができます．

また，転送モードは，マニュアル転送，ハード転送（プログラム転送，DMA転送）があります．

マニュアル転送は，SCSI上のACK/REQ信号をソフトによって制御して転送します．プログラム転送は，コントローラのコマンドで行い，DMA転送は，コントローラにDMAモードを設定することで行います．

図1にMB89352のピン配置を，図2に内部ブロック図を示します．表2に各ピンの説明を示します．図3，図4，図5に，リード/ライト，DMAのタイミングを示します．

このコントローラは，14個のレジスタによって制御されます．表3に内部レジスタを示します．

◈ レジスタの説明

レジスタの選択には，A_0～A_3の4ビットのアドレスで行います．各々のレジスタは，リード/ライトが可能ですが，動作が異なります．

● BDIDレジスタ (0)

ライト時は，SCSI上でのバス・デバイスID（0～7）を2進数で指定します．リード時は，バス・デバイスIDを示します．ビット0～7上の1ビットがONとな

〈表1〉[9] 現在市販されているSCSI用LSIの主な機能

メーカ名	ウェスタンデジタル	ウェスタンデジタル	日本エヌシーアール	富士通	アダプテック
型　名	WD33C93A	WD33C93	NCR5380S	MB87030/31*	AIC6250
同期転送，非同期転送	可	可	可	可	可
最大転送速度	5Mバイト/秒	4Mバイト/秒	2Mバイト/秒	4Mバイト/秒	5Mバイト/秒
アービトレーション	可	可	可	可	可
転送カウンタ(24ビット)	有	有	有	有	有
ドライバ/レシーバ	内蔵	内蔵	外付け	外付け	内蔵
データ・バッファリング	12バイトFIFO	5バイトFIFO	2バイトFIFO	8バイトFIFO	8バイトFIFO
データ・アクセス・モード	P-I/O DMA バーストDMA WD-バス	P-I/O DMA WD-バス	P-I/O DMA	バーストDMA	P-I/O DMA
オフセット	1～12	1～5	1	1～8	1～8
製造プロセス	C-MOS	C-MOS	NMOS(C-MOS)	C-MOS	C-MOS
パッケージ	40ピンDIP 44ピンPLCC	40ピンDIP 44ピンPLCC	48ピンDIP 44ピンPLCC	88ピンPGA 100ピンQFP	48ピンDIP 52ピンPLCC

＊ バス・ドライバがつき改良されたのがMB89352．詳しくは§4-2に解説．新たにMB87033も発表された

〈表 2〉[1] MB89352のピン機能

信号名	入出力	ピン番号	機　　能
RST	入　力	1	コントローラのリセット入力
CS	入　力	2	コントローラのセレクト入力
A_0 A_1 A_2 A_3	入　力	3 4 5 6	コントローラの内部レジスタをセレクトするためのアドレス入力
DP_0	出　力	19	D_7～D_0の奇数パリティ出力
D_P D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	入　力	20 29 28 27 26 25 23 22 21	奇数パリティ・ビット． 8ビットの双方向性3ステートのデータ・バス． RD, WRが"H"のときハイ・インピーダンスになる
RD	入　力	43	内部レジスタを読み出すためのストローブ信号．CS＝"L"で有効となる
WR	入　力	44	内部レジスタを書き込むためのストローブ信号．CS＝"L"で有効となる
CLK	入　力	47	コントローラの制御のためのクロック
INTR	出　力	30	コマンドの終了，またはエラーが発生したことを通知するための信号

信号名	入出力	ピン番号	機　　能
DACK	入　力	46	DMA要求に対する応答信号
DREQ	出　力	45	DMA転送の要求信号
SD_7 SD_6 SD_5 SD_4 SD_3 SD_2 SD_1 SD_0 SD_P	入出力	7 8 10 11 13 14 15 17 18	SCSI上のデータ・バス 負論理信号
SEL BSY I/O C/D MSG REQ ACK ATN SRST	入出力	34 32 35 36 38 39 41 42 31	SCSI上の制御信号であり，負論理
V_{CC}	入　力	48	＋5V電源入力
V_{SS}		24	0V（GND）
DG		9 16 33 40	ドライバ・グラウンド． V_{SS}と同等

ります．

● **SCTLレジスタ** (1)

コントローラの動作モードを設定します．コントローラのリセット，各フェーズの動作を決定します(図6参照)．

〈図 1〉[1] MB89352のピン配置

● **SCMDレジスタ** (2)

ライトされたコマンドを実行します．またはコントローラとCPUとの転送モードを設定します．

コマンドを書き込みコントローラが実行可能の状態であれば，そのコマンドを実行します(図7)．

以下のコマンドがあります．

(1) バス・リリース

SCSIバスを解放します．

(2) セレクト

セレクション，リセレクションを行います．

(3) リセットATN

ATN信号をリセットします．

(4) セットATN

ATN信号をセットします．

(5) トランスファ

データの転送を行います．

(6) トランスファ・ポーズ

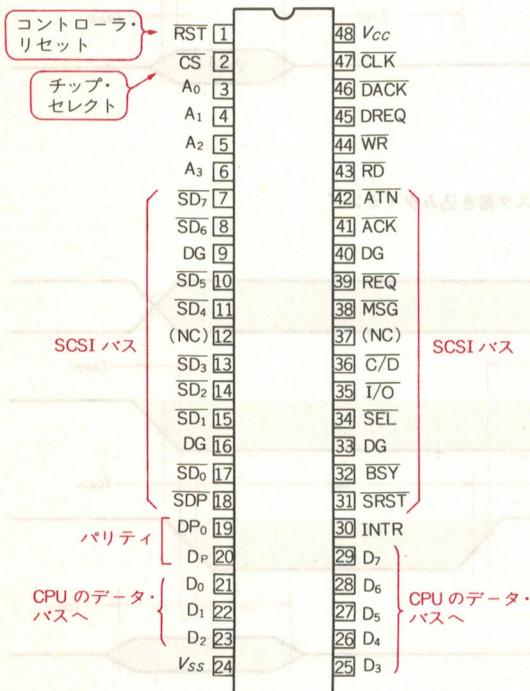
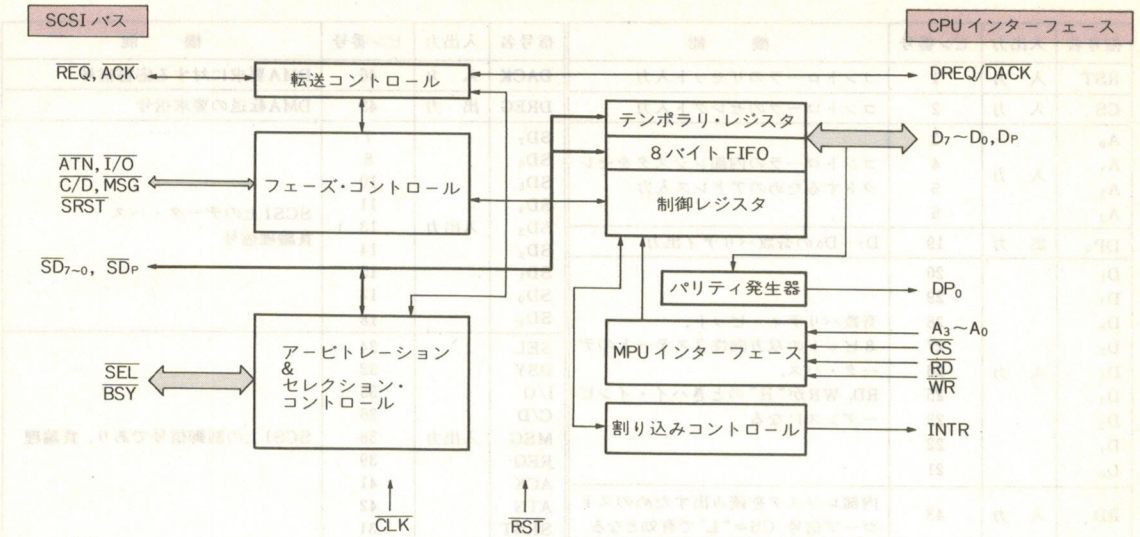
ターゲット動作のとき，ハード転送を一時停止させます．

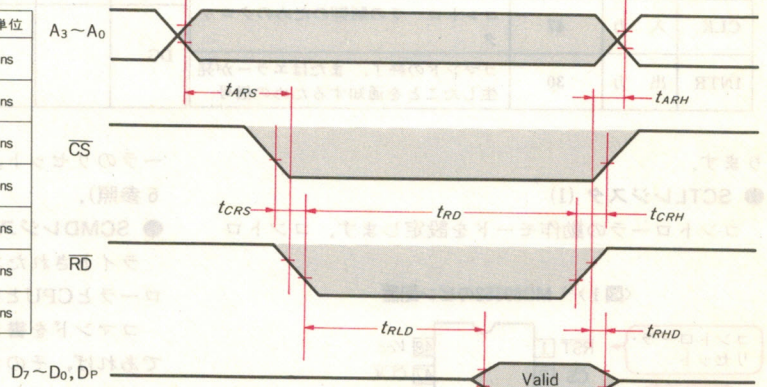
(7) リセットACK/REQ

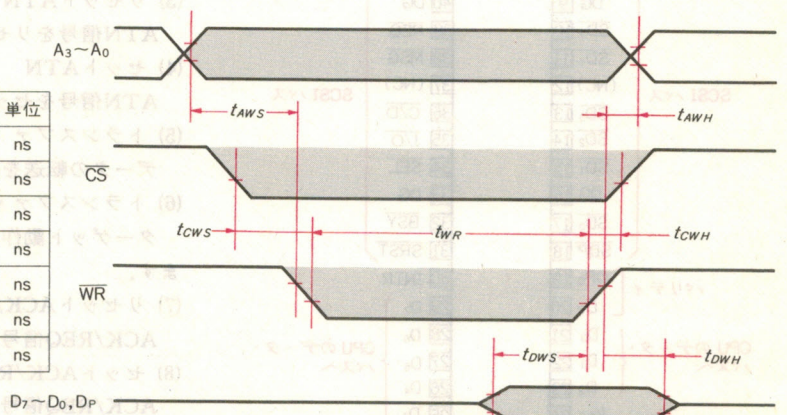
ACK/REQ信号をリセットします．

(8) セットACK/REQ

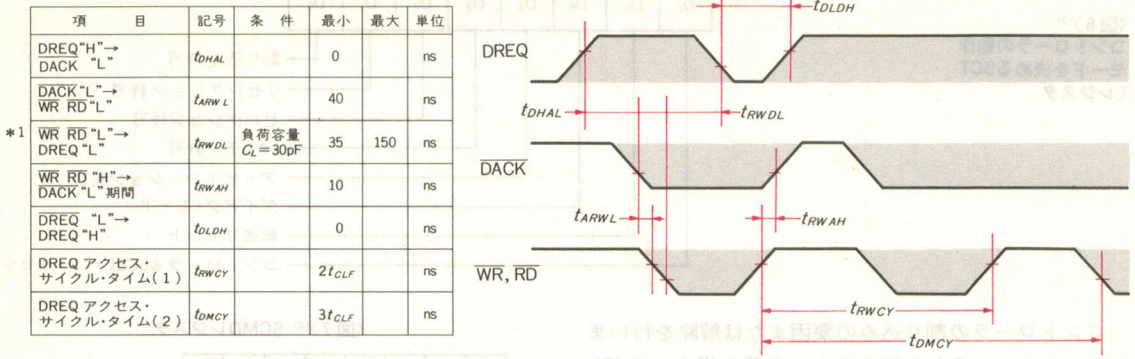
ACK/REQ信号をセットします．

● **INTSレジスタ** (4)

〈図2〉[1] MB89352の内部ブロック図

〈図3〉[1] MB89352のレジスタ読み出しタイミング

項目	記号	条件	最小	最大	単位
アドレス・セットアップ	t_{ARS}		40		ns
アドレス・ホールド	t_{ARH}		10		ns
$\overline{CS}$ セットアップ	t_{CRS}		25		ns
$\overline{CS}$ ホールド	t_{CRH}		10		ns
$\overline{RD}$ パルス幅	t_{RD}		120		ns
$\overline{RD}$ "L"→データ確定	t_{RLD}	負荷容量 C_L=80pF		90	ns
$\overline{RD}$ "H"→データ・ホールド	t_{RHD}	負荷容量 C_L=20pF	10	60	ns

〈図4〉[1] MB89352のレジスタ書き込みタイミング

項目	記号	最小	単位
アドレス・セットアップ	t_{AWS}	40	ns
アドレス・ホールド	t_{AWH}	10	ns
$\overline{CS}$ セットアップ	t_{CWS}	25	ns
$\overline{CS}$ ホールド	t_{CWH}	10	ns
データ・バス・セットアップ	t_{DWS}	30	ns
データ・バス・ホールド	t_{DWH}	20	ns
$\overline{WR}$ パルス幅	t_{WR}	100	ns

〈図5〉(1) MB89352のDMAアクセスのタイミング

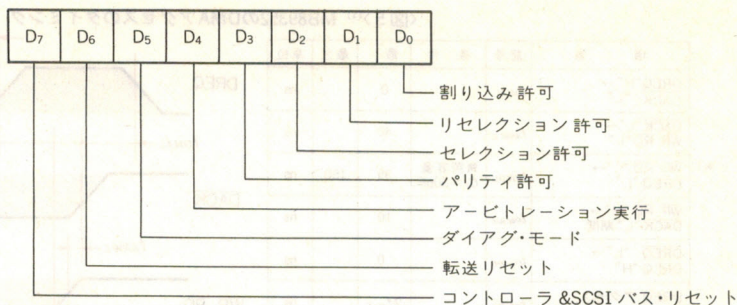
	項　　目	記号	条　件	最小	最大	単位
	$\overline{\text{DREQ}}$"H"→ $\overline{\text{DACK}}$ "L"	t_{DHAL}		0		ns
	$\overline{\text{DACK}}$"L"→ $\overline{\text{WR}}$ $\overline{\text{RD}}$"L"	t_{ARWL}		40		ns
＊1	$\overline{\text{WR}}$ $\overline{\text{RD}}$"L"→ DREQ"L"	t_{RWDL}	負荷容量 C_L＝30pF	35	150	ns
	$\overline{\text{WR}}$ $\overline{\text{RD}}$"H"→ $\overline{\text{DACK}}$"L"期間	t_{RWAH}		10		ns
	$\overline{\text{DREQ}}$ "L"→ DREQ"H"	t_{DLDH}		0		ns
	DREQアクセス・サイクル・タイム(1)	t_{RWCY}		$2t_{CLF}$		ns
	DREQアクセス・サイクル・タイム(2)	t_{DMCY}		$3t_{CLF}$		ns

DREQ
$\overline{\text{DACK}}$
$\overline{\text{WR}}$, $\overline{\text{RD}}$
t_{DLDH}　t_{DHAL}　t_{RWDL}　t_{ARWL}　t_{RWAH}　t_{RWCY}　t_{DMCY}

(＊1) ライト時(アウトプット時)，データ・バッファ・レジスタがFullになるときに適用する．
　　　リード時(インプット時)，データ・バッファ・レジスタがEmptyになるときに適用する．
(＊2) $\overline{\text{WR}}$,$\overline{\text{RD}}$パルス幅は，レジスタ書き込みタイミング，読み出しタイミングの規定に従う．

〈表3〉(2) 内部レジスタ

アドレス(16進数)	名　称(略称)	R/W	ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	パリティ
			レ　ジ　ス　タ・ビ　ッ　ト								
0	Bus Device ID (BDID)	R	＃7	＃6	＃5	＃4	＃3	＃2	＃1	＃0	"0"
		W			—			ID4	ID2	ID1	—
1	Spc Control (SCTL)	R/W	Reset & Disable	Control Reset	Diag Mode	ARBIT Enable	Parity Enable	Select Enable	ReSelect Enable	INT Enable	P
2	Command (SCMD)	R/W	Command Code			RST Out	Intercept Xfer	Transfer Modifier PRG Xfer	"0"	Term Mode	P
4	Interrupt Sense (INTS)	R	Selected	Re-Selected	Dis-Connected	Command Complete	Service Required	Time Out	SPC Hard Error	Reset Condition	P
		W	Reset Interrupt								—
5	Phase Sense (PSNS)	R	REQ	ACK	ATN	SEL	BSY	MSG	C/D	I/O	P
	SPC Diag. Control (SDGC)	W	Diag REQ	Diag ACK	Xfer Enable	—	Diag BSY	Diag MSG	Diag C/D	Diag I/O	—
6	SPC Status (SSTS)	R	Connected INIT	TARG	SPC BSY	Xfer in Progress	SCSI RST	TC＝0	DREG Status FULL	EMPTY	P
7	SPC Error Status (SERR)	R	Data SCSI	Error SPC	Xfer Out	"0"	TC P-Error	"0"	Short Period	"0"	P
8	Phase Control (PCTL)	R/W	Bus Free Interrupt Enable	"0"				Transfer MSG Out	Phase C/D Out	I/O Out	P
9	Modified Byte Counter (MBC)	R	"0"				MBC ビット3	ビット2	ビット1	ビット0	P
A	Data Register (DREG)	R	Internal Data Register (8バイトFIFO)								P
		W	ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	
B	Temporary Register (TEMP)	R	Temporary Data (Input：From SCSI) ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	P
		W	Temporary Data (Output：To SCSI) ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	P
C	Transfer Counter High(TCH)	R	Transfer Counter High (MSB)								P
		W	ビット23	ビット22	ビット21	ビット20	ビット19	ビット18	ビット17	ビット16	
D	Transfer Counter Mid(TCM)	R	Transfer Counter Middle (2nd Byte)								P
		W	ビット15	ビット14	ビット13	ビット12	ビット11	ビット10	ビット9	ビット8	
E	Transfer Counter Low(TCL)	R	Transfer Counter Low (LSB)								P
		W	ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0	

(注1) 書き込み動作において―が記入されているビットは，"1"，"0"のどちらかを書き込んでもよいことを示す．
(注2) 読み出し専用レジスタへの書き込み動作は無視される．
(注3) 読み出し時において"0"と示されているビットは必ず"0"が読み出される．

〈図6〉[2]
コントローラの動作モードを決めるSCTLレジスタ

コントローラの割り込みの要因または解除を行います．コントローラから割り込みが必要な場合，INTR信号を出力しますが，SCTLレジスタによりマスク可能です．割り込み解除は，割り込み要因の対応するビットに“1”を書き込むことにより行います．

割り込み原因には，SCSIバス上でセレクション，リセレクションが行われたとき，セレクト，転送コマンドが終了したときなどがあります．

本レジスタをリセットしないで，コマンドを発行しても正常に動作は行われません．

● PSNSレジスタ (5)

ダイアグ・モード(テスト・モード)の場合，SCSI上の制御線をコントロールします．それ以外では，SCSI上の制御信号の状態を示します．

これによってSCSIの制御線の確認ができます．このレジスタは，コントローラの状態にかかわらずリードできます．

● SDGCレジスタ (5)

ダイアグ・モードのとき，コントローラを疑似的にコントロールするレジスタです．

● SSTSレジスタ (6)

コントローラの内部状態を示すレジスタで，常にリード可能です．

● SERRレジスタ (7)

コントローラ内部で検出されたエラーの内容を示します．

● PCTLレジスタ (8)

実行する転送フェーズを指定します．またはセレクション，リセレクションの区別を行います．

● MBCレジスタ (9)

プログラムまたはDMA転送モードでの，データ転送バイトを示すカウンタです．

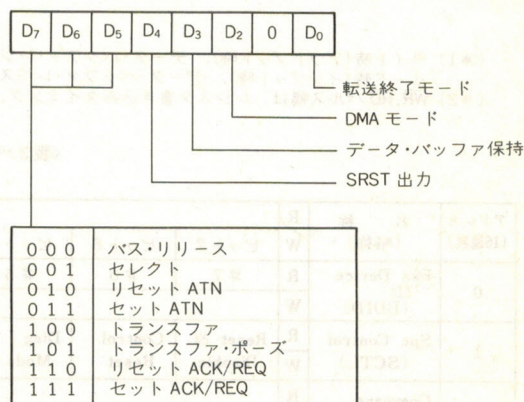
● DREGレジスタ (10)

コントローラ内部の転送用データ・レジスタです．8バイトのFIFOから構成されています．ハード転送の場合このレジスタを使用します．

● TEMPレジスタ (11)

ハード転送以外で転送する場合に使用します．

● TCH, TCM, TCLレジスタ (12, 13, 14)

〈図7〉[2] SCMDレジスタ

3バイトから成るレジスタで，ハード転送時カウント・ダウンされて，転送の残りバイトを示します．それ以外に，セレクト時のタイム・アウト値を設定します．

◆ 動作の説明

コントローラは，これらのレジスタによってSCSIのコントロールを行います．そのコントロールは，SCMDレジスタのコマンドで行います．

● バス・フリー

ターゲット動作中に，バス・フリー・フェーズへの移行を要求します．またバス・フリー待ちのセレクト・コマンドを解除します．

● セレクト

イニシエータ，ターゲットの結合を開始します．もしバス・フリー状態でなければ，バス・フリーが検出されるのを待ちます．

SCTLレジスタのアービトレーション＝1の場合は，アービトレーション・フェーズを行い，バスの占有権を獲得します．バス獲得ができなかったとき，コマンドは終了します．

アービトレーション＝0の場合は，すぐにセレクションを開始します．セレクションは，PCTLレジスタのビット0(I/O)＝0のときセレクション・フェーズを，ビット0＝1のときリセレクション・フェーズを

実行します．

セレクション・フェーズにおいてANT信号が必要なときは，SET ANTコマンドをあらかじめ発行しておきます．

TEMPレジスタには，結合するイニシエータまたはターゲットのID番号を入れておきます．

TCH, TCMレジスタは，BSY信号の時間監視の値をセットします．この時間は，セレクション，リセレクション・フェーズ開始してから，BSY信号がアクティブになるまでです．

N（TCH：High，TCM：Low）

$N \neq 0$の場合　時間$=(N \times 256+15) \times T_{clf} \times 2$

$N=0$の場合　時間$=\infty$

（T_{clf}はコントローラのクロック周期）

TCLレジスタは，バス・フリー後アービトレーション，セレクション・フェーズを開始するための時間をセットします．値は0～15までの値です．

時間$=(\mathrm{TCL}+6) \times T_{clf} \sim (\mathrm{TCL}+7) \times T_{clf}$

コントローラのクロックが8 MHzのとき，TCL＝4で1.25μsとなります．

● **セット ATN**

イニシエータとして動作する場合のみ有効となります．

結合前にコマンドが出された場合は，セレクション・フェーズ実行時にATN信号がアクティブになります．結合中であればコマンドが出された時点で，ATN信号がアクティブとなります．

● **リセット ATN**

アクティブのATN信号をインアクティブにします．

ただし，SCSIのバス結合が解除された場合，本コマンドが発行しなくてもリセットされます．

● **トランスファ**

各転送フェーズを実行させます（コマンド，データ，ステータス，メッセージ・フェーズ）．

このコマンドで実行される転送モードは，ハード転送と呼び，コントローラによって転送シーケンスの制御が行われます．転送は，レジスタによって制御します．

TCH, TCM, TCLレジスタは，転送バイト数を24ビットで指定します．

PCTLレジスタは，転送フェーズを3ビットで指定します．

000：データ・アウト・フェーズ

001：データ・イン・フェーズ

010：コマンド・フェーズ

011：ステータス・フェーズ

110：メッセージ・アウト・フェーズ

111：メッセージ・イン・フェーズ

コントローラは，PCTLのフェーズとバス実行フェーズが同じとき転送を開始します．もしフェーズが一致しない場合は，サービス要求割り込みが発生します．

このときは，フェーズを合わせて再度転送を行うか，マニュアルで転送を行います．

● **トランスファ ポーズ**

ターゲット動作しているとき，ハード動作を中止させます．

● **セット ACK/REQ**

マニュアル転送を行うためコマンドです．イニシエータ動作時は，ACK信号をターゲット動作時は，REQ信号をアクティブにします．

マニュアル転送のときは，TEMPレジスタでデータのやり取りをします．ハード転送と同様にPCTLレジスタに転送フェーズをセットしておきます．

● **リセット ACK/REQ**

セット ACK/REQコマンドでアクティブにした信号をリセットします．

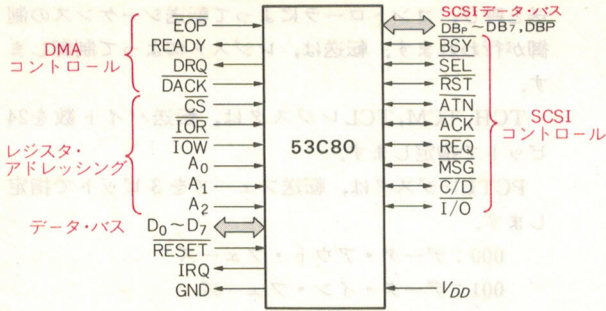
イニシエータ動作時はACK信号を，ターゲット動作時はREQ信号をインアクティブにします．

〈表4〉富士通製SCSIコントローラ一覧

	MB87033	MB87030/31	MB89352	MB89351
シングルエンド・ドライバ/レシーバ	内　蔵	外付け	内　蔵	外付け
ディファレンシャル対応	不　可	可	不　可	可
同期転送	可	可	不　可	不　可
転送バイト・カウンタ	28ビット	24ビット	24ビット	24ビット
アービトレーション失敗時割り込み	あ　り	な　し	な　し	な　し
アテンション・コンディション検出割り込み	あ　り	な　し	な　し	な　し
FIFO Full/Empty 割り込み	な　し	な　し	あ　り	あ　り
割り込み信号数	2	1	1	1
CPUバス・パリティ・ジェネレータ	あ　り	な　し	あ　り	あ　り
データ転送バス	独　立	独　立	CPUバスと共通	CPUバスと共通

〈図8〉機能別ピン配置

〈図9〉[1] ピン配置

信号	ピン	53C80	ピン	信号
$\overline{DB_7}$	1		48	$\overline{DB_6}$
$\overline{RST}$	2		47	$\overline{DB_5}$
GND	3		46	GND
$\overline{BSY}$	4		45	$\overline{DB_4}$
$\overline{SEL}$	5		44	$\overline{DB_3}$
$\overline{ATN}$	6		43	$\overline{DB_2}$
NC	7		42	NC
$\overline{RESET}$	8		41	$\overline{DB_1}$
IRQ	9		40	$\overline{DB_0}$
DRQ	10		39	GND
$\overline{EOP}$	11		38	$\overline{DB_P}$
$\overline{DACK}$	12		37	$\overline{REQ}$
GND	13		36	$\overline{ACK}$
READY	14		35	$\overline{I/O}$
A_0	15		34	GND
A_1	16		33	$\overline{C/D}$
A_2	17		32	$\overline{MSG}$
NC	18		31	NC
$\overline{CS}$	19		30	D_0
$\overline{IOW}$	20		29	D_1
$\overline{IOR}$	21		28	D_2
D_7	22		27	D_3
D_6	23		26	D_4
D_5	24		25	V_{DD}

なお，富士通製のSCSIコントローラの比較を表4に示します．

NCR53C80

次に，市場に出るのが早く，利用の多いNCRの53C80について説明します．

図8に機能別に分けたピン配置，図9に53C80のピン配置，表5に各ピンの詳しい説明を示します．また，図10に内部ブロック図を示します．

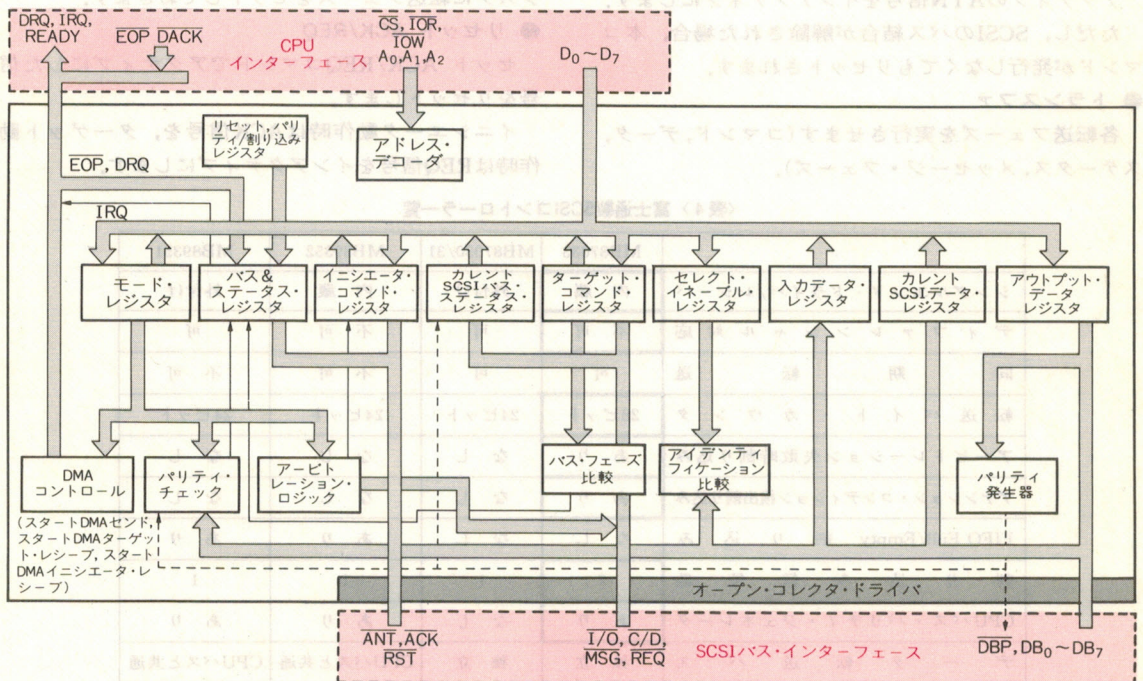
◈ 内部レジスタ

53C80は，CPUから見ると八つのレジスタの集合体としてみることができます．プログラミングは，この八つのレジスタを駆使して行うことになります（表6）．

● カレントSCSIデータ・レジスタ（R）アドレス＝0

プログラムI/Oで，CPUがSCSIデータ・バスを読み込むときと，アービトレーション中に，ほかのプライオリティの高いデバイスが，アービトレーションを行っていないかどうかをチェックするときに使用します．

モード・レジスタで，イネーブル・パリティ・チェック・ビットが1にセットされているときは，このレジスタのリード・サイクルの最初で，パリティがチェックされます．パリティ・エラーのときは，バス/ステータス・レジスタのパリティ・エラー・ビットが1に

〈図10〉[11] 53C80の内部ブロック図

〈表 5〉 53C80のピン説明

信号名	ピン番号	入出力	説　　明
DMAインターフェース			
$\overline{EOP}$	11	入力	End Of Process. DMA転送中に$\overline{EOP}$=“L”とすると，DMA転送が終了する
READY	14	出力	ブロック・モードDMAでの1バイトごとのDMA要求を表す
DRQ	10	出力	DMAリクエスト．モード・レジスタのDMAモード・ビット＝1で，データ・レジスタ中にデータが用意されたときに出力される． ノーマル・モードのDMAでは，DMA/$\overline{DACK}$で1バイトごとのハンドシェイクを行う
$\overline{DACK}$	12	入力	DMAアクノレッジ．$\overline{DACK}$=“L”によってDRQがリセット(＝“H”)され，データ・レジスタが入出力のためにセレクトされる
CPUインターフェース			
$\overline{CS}$	19	入力	チップ・セレクト．A_2～A_0で指定された内部レジスタのリード/ライトをイネーブルする
$\overline{IOR}$	21	入力	I/Oリード・ストローブ．$\overline{CS}$とA_2～A_0で指定されたレジスタを読み込む． また，$\overline{DACK}$と共に使われると($\overline{DACK}$=“L”，$\overline{IOR}$=“L”)，インプット・データ・レジスタがセレクトされる(DMA転送)
$\overline{IOW}$	20	入力	I/Oライト・ストローブ．$\overline{CS}$とA_2～A_0で指定されたレジスタに書き込む． また，$\overline{DACK}$と共に使われると($\overline{DACK}$=“L”，$\overline{IOW}$=“L”)，アウトプット・データ・レジスタがセレクトされる(DMA転送)
A_2 A_1 A_0	17 16 15	入力	CS, IOR, IOW と共に使われ，内部レジスタのセレクトを行う
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	22 23 24 26 27 28 29 30	入出力 3ステート	データ・バス
IRQ	9	出力	インタラプト・リクエスト
$\overline{RESET}$	8	入力	すべての内部レジスタをクリアする(ただし，SCSIバス上には，$\overline{RST}$は出力しない)
電源			
V_{DD}	25		＋5Vサプライ
GND	3,13,34, 39,46		グラウンド．ノイズ・マージン改善のために，5本のグラウンドが用意されている
SCSIインターフェース			
$\overline{ATN}$ $\overline{BSY}$ $\overline{ACK}$ $\overline{RST}$ $\overline{MSG}$ $\overline{SEL}$ $\overline{C/D}$ $\overline{REQ}$ $\overline{I/O}$ $\overline{DB_7}$ $\overline{DB_6}$ $\overline{DB_5}$ $\overline{DB_4}$ $\overline{DB_3}$ $\overline{DB_2}$ $\overline{DB_1}$ $\overline{DB_0}$ $\overline{DB_P}$	6 4 36 2 32 5 33 37 35 1 48 47 45 44 43 41 40 38		SCSIバス信号

セットされます．アービトレーション中には，パリティ・ビットは常にフォールスです．

7	6	5	4	3	2	1	0
$\overline{DB_7}$	$\overline{DB_6}$	$\overline{DB_5}$	$\overline{DB_4}$	$\overline{DB_3}$	$\overline{DB_2}$	$\overline{DB_1}$	$\overline{DB_0}$

〔カレントSCSIデータ・レジスタ〕

● アウトプット・データ・レジスタ (W) アドレス＝0

プログラムI/O, DMA I/Oの両方で，SCSIバスにデータを出力するときに使用します．

プログラムI/Oのときは，$\overline{CS}$＝“L”，A_2＝“L”，A_1＝“L”，A_0＝“L”，$\overline{IOW}$＝“L”でセレクトされ，DMA I/Oのときは，$\overline{DACK}$＝“L”，$\overline{IOW}$＝“L”，$\overline{CS}$＝“H”，A_2～A_0＝× でセレクトされます．また，アービトレーション，セレクション，リセレクション時のIDビットを出力するときにも使用されます．

(アウトプット・データ・レジスタ)

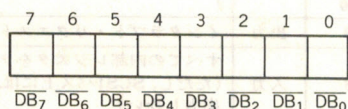
〈表 6〉 53C80の内部レジスタ

A_2 A_1 A_0	R/W	レ　ジ　ス　タ　名
0 0 0	R W	カレントSCSIデータ・レジスタ アウトプット・データ・レジスタ
0 0 1	R/W	イニシエータ・コマンド・レジスタ
0 1 0	R/W	モード・レジスタ
0 1 1	R/W	ターゲット・コマンド・レジスタ
1 0 0	R W	カレントSCSIバス・ステータス・レジスタ セレクト・イネーブル・レジスタ
1 0 1	R W	バス/ステータス・レジスタ スタートDMAセンド・レジスタ
1 1 0	R W	インプット・データ・レジスタ スタートDMAターゲット・レシーブ・レジスタ
1 1 1	R W	リセット・パリティ/インタラプト/レジスタ スタートDMAイニシエータ・レシーブ・レジスタ

● インプット・データ・レジスタ (R) アドレス＝6

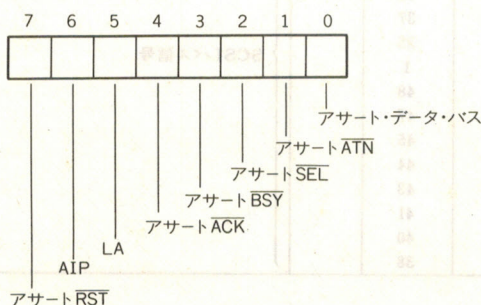
DMA I/Oで，SCSIバス上のデータをラッチします．$\overline{IOR}$＝“L”，$\overline{DACK}$＝“L”，$\overline{CS}$＝“H”，A_2～A_0＝× でセレクトされます．また，CPUからも，$\overline{IOR}$＝“L”，$\overline{CS}$＝“L”，A_2＝“H”，A_1＝“H”，

A_0＝“L” でセレクトすることができます．

このレジスタにデータがラッチされるのは，モード・レジスタでDMAモードに設定され，DMAターゲット・レシーブ中の，$\overline{ACK}$の立ち下がり(⏋⎿)と，DMAイニシエータ・レシーブ中の，$\overline{REQ}$の立ち下がり(⏋⎿)のときです．

〔インプット・データ・レジスタ〕

● イニシエータ，コマンド・レジスタ（R/W）アドレス＝1

イニシエータに関係の深いSCSIバス信号($\overline{ACK}$, $\overline{BSY}$, $\overline{SEL}$, $\overline{ATN}$, $\overline{RST}$)をアサート(出力)したり，アービトレーションの状態をモニタするときに使用します．

● 読み込み時

リード時に重要なのは，LAビットとAIPビットの二つで，ほかのビットは，たんにそのビットの状態をリード・バックしているにすぎません．

▶LA(Lost Arbitration)ビット

アービトレーションに負けたかどうかをモニタします．このビットは，モード・レジスタのアービトレート・ビットに1が書き込まれてから意味をもつようになり，LA＝1であるということは，次の状態を意味します．

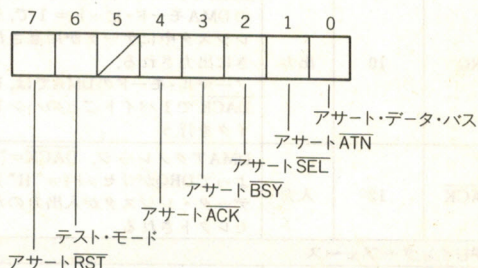
(1) バス・フリーを検出し，
(2) $\overline{BSY}$＝“L” にし，
(3) SCSIデータ・バスにIDを出力したが，
(4) ほかのデバイスが，$\overline{SEL}$＝“L” にしたので，アービトレーションに負けた．

▶AIP(Arbitration In Progress)ビット

アービトレーションが，進行中かどうかをモニタします．このビットもモード・レジスタのアービトレート・ビットに1が書き込まれてから意味をもつようになり，AIP＝1であるということは，次の状態を意味します．

(1) バス・フリーが検出され，
(2) $\overline{BSY}$＝“L” にし，
(3) アウトプット・データ・レジスタの内容をSCSIデータ・バスに出力している．

この状態は，モード・レジスタのアービトレート・ビットがリセットされるまで変化しません．

● 書き込み時

▶ビット7＝アサート$\overline{RST}$

1を書き込むと，SCSIバス上の$\overline{RST}$＝“L” となります．この状態($\overline{RST}$＝“L”)は，このビットに0を書くか，またはチップがリセット($\overline{RESET}$＝“L”)されるまで，解除されません．

同時にIRQ(9ピン)もアサートされ，インタラプト・ラッチと，このビットを除くすべての内部ロジック，コントロール・レジスタはリセットされます．

▶ビット6＝テスト・モード

チップのメインテナンス/テスト用のビットで，1を書き込むと，すべての出力ドライバをディセーブルします．0にもどすと，通常のオペレーションになります．

▶ビット4＝アサート$\overline{ACK}$

1で$\overline{ACK}$＝“L”，0で$\overline{ACK}$＝“H” にします．$\overline{ACK}$＝“L” にするためには，イニシエータとして(モード・レジスタのターゲット・モード・ビット＝0)，設定されていなければなりません．

▶ビット3＝アサート$\overline{BSY}$

1で$\overline{BSY}$＝“L”，0で$\overline{BSY}$＝“H”にします．

▶ビット2＝アサート$\overline{SEL}$

1で$\overline{SEL}$＝“L”，0で$\overline{SEL}$＝“H”にします．

▶ビット1＝アサート$\overline{ATN}$

1で$\overline{ATN}$＝“L”，0で$\overline{ATN}$＝“H” にします．ビット4と同様$\overline{ATN}$＝“L” にするためには，イニシエータとして設定されていることが必要です．

▶ビット0＝アサート・データ・バス

1を書き込むと，アウトプット・データ・レジスタの内容を，SCSIデータ・バスに出力します．パリティもジェネレートされ，$\overline{DB_P}$に出力されます．

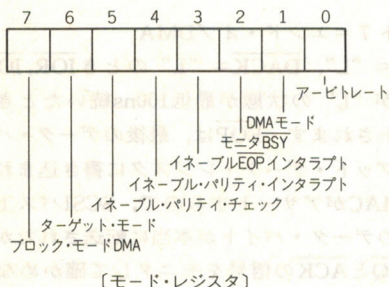
このチップがイニシエータのときは，以下の条件がそろっていないとデータは出力されないので，注意が必要です．

(1) モード・レジスタのターゲット・モード・ビット＝0
(2) SCSIバスの$\overline{\text{I/O}}$＝“H”
(3) ターゲット・コマンド・レジスタ中の下位3ビットが，SCSIバス上の，$\overline{\text{C/D}}$, $\overline{\text{I/O}}$, $\overline{\text{MSG}}$にそれぞれマッチしていること

また，DMAセンドのときも，このビットが1に設定されていなければなりません．

● **モード・レジスタ（R/W）アドレス＝2**

チップのモードを設定します．

〔モード・レジスタ〕

▶ビット7＝ブロック・モードDMA

DMAのDRQ/DACKのハンドシェイクの方法を設定します．DMAモード・ビットが1で，このビットが0のときは，通常のインターロック・ハンドシェイクを行い，1バイトごとに，DRQと$\overline{\text{DACK}}$がハンドシェイクを行います．

DMAモード・ビットが1で，このビットも1のときには，ブロック・モードDMAと呼ばれ，$\overline{\text{DACK}}$は複数バイトにわたって常に“L”です．$\overline{\text{IOR}}$または$\overline{\text{IOW}}$のパルスで，1バイトの転送が行われ，READYピン(ピン14)が，次の1バイトの転送要求を行います．

▶ビット6＝ターゲット・モード

1でターゲット，0でイニシエータとしてこのチップを設定します．$\overline{\text{ATN}}$, $\overline{\text{ACK}}$をアサートするときは0，$\overline{\text{C/D}}$, $\overline{\text{I/O}}$, $\overline{\text{MSG}}$, $\overline{\text{REQ}}$をアサートするときは1にそれぞれ設定します．

▶ビット5＝イネーブル・パリティ・チェック

パリティ・エラーが発生したときに，その情報をバス/ステータス・レジスタのパリティ・エラー・ビットにセーブするか，無視するかを決めます．1でセーブ，0で無視です．

▶ビット4＝イネーブル・パリティ・インタラプト

1にセットすると，パリティ・エラーが発生したときに，インタラプト(IRQ)を発生します．このインタラプトを発生させるには，ビット5のイネーブル・パリティ・チェック・ビットも1にセットしておかなければなりません．

▶ビット3＝イネーブルEOPインタラプト

1にセットすると，DMACからの$\overline{\text{EOP}}$(ピン11)＝“L”で，インタラプトを発生します．

▶ビット2＝モニタ$\overline{\text{BSY}}$

1にセットすると，SCSIバス上の$\overline{\text{BSY}}$が“H”になってはいけない所で“H”になったときに，インタラプトを発生します．このインタラプトが発生すると，イニシエータ・コマンド・レジスタの下位6ビットがクリアされ，すべての信号がSCSIバスから取り除かれます．

▶ビット1＝DMAモード

DMA転送を行う前にセットすべきもので，スタートDMAセンド・レジスタ，スタートDMAターゲット・レシーブ・レジスタ，スタートDMAイニシエータ・レシーブ・レジスタに書き込む前に設定します．また，ターゲット・モード・ビットも，スタートDMAイニシエータ・レシーブを行うときには0，スタートDMAターゲット・レシーブを行うときには1にセットします．

さらに，スタートDMAセンドを行うときには，イニシエータ・コマンド・レジスタのアサート・データ・バス・ビットを1にセットしておかなければなりません．なお，このビットは，SCSIバスで$\overline{\text{BSY}}$＝“L”になっているときでないとセットできません．

《重要》

DMAモードでは，REQ/ACKのハンドシェイクは，自動的に行われます．また，$\overline{\text{CS}}$と$\overline{\text{DACK}}$が同時に“L”になることは許されません．

▶ビット0＝アービトレート

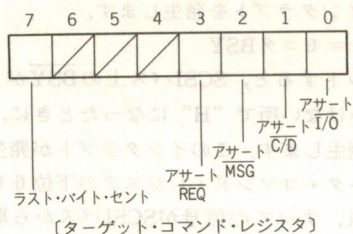
アウトプット・データ・レジスタにIDを書いたのち，このビットを1にセットすると，次のプロセスを行います．

(1) バス・フリーが検出されるまで待つ
(2) IDを出力し，$\overline{\text{BSY}}$＝“L”とする
(3) アービトレーションの結果を，イニシエータ・コマンド・レジスタのAIP, LA各ビットにセットする

● **ターゲット・コマンド・レジスタ（R/W）アドレス＝3**

ターゲットとして接続された場合(モード・レジスタのターゲット・モード・ビット＝1)，ターゲットの動作を行う信号($\overline{\text{I/O}}$, $\overline{\text{C/D}}$, $\overline{\text{MSG}}$, $\overline{\text{REQ}}$)をコントロールします．また，イニシエータとして接続した場合は，データをSCSIバスに出力するためには，このレジスタの下位3ビットは，カレントSCSIバス・ステータス・レジスタの$\overline{\text{MSG}}$, $\overline{\text{C/D}}$, $\overline{\text{I/O}}$の各ビットと一致していなければなりません．

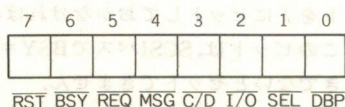
さらに，イニシエータでかつ，DMAモードのときは，SCSIバス上の$\overline{\text{MSG}}$, $\overline{\text{C/D}}$, $\overline{\text{I/O}}$の各信号が，これらのビットと一致していないと，$\overline{\text{REQ}}$＝“L”になったときに，フェーズ・ミスマッチのインタラプトが発生します．

〔ターゲット・コマンド・レジスタ〕

▶ビット7＝ラスト・バイト・セント（R）

DMAで，最後のバイトがSCSIバス上に出力され，それがREQ/ACKによって転送され終わったときに，セットされます．

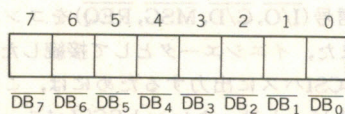
● カレントSCSIバス・ステータス・レジスタ（R）アドレス＝4

七つのSCSIバスの制御信号線と，パリティ・ビットをモニタします．現在のバスのフェーズがどうなっているか，$\overline{REQ}$を出しているターゲットがいないか，また，インタラプトが発生したときに，何が原因かをチェックする目的で使用されます．

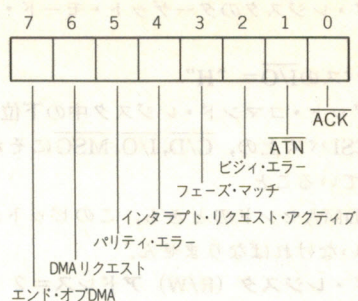
7	6	5	4	3	2	1	0
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

〔カレントSCSIバス・ステータス・レジスタ〕

● セレクト・イネーブル・レジスタ（W）アドレス＝4

セレクション/リセレクション・フェーズで，自分のIDがセレクト/リセレクトされたときに，インタラプトを発生できます．すなわち，このレジスタに書き込んだIDビットと同じIDがSCSIバスに現れ，かつ，$\overline{BSY}$＝“H”，$\overline{SEL}$＝“L”という状態が，最低400ns以上続いたときにインタラプトを発生します．このインタラプトは，0を書き込むことでディセーブルできます．

なお，モード・レジスタのイネーブル・パリティ・チェック・ビットが1にセットされている場合は，セレクション/リセレクション中にもパリティ・ビットがチェックされます．

〔セレクト・イネーブル・レジスタ〕

● バス/ステータス・レジスタ（R）アドレス＝5

ここでは，カレントSCSIバス・ステータス・レジスタに入りきらなかったSCSIバスの制御信号線（$\overline{ATN}$，$\overline{ACK}$）と，その他のステータスをモニタすることができます．

〔バス/ステータス・レジスタ〕

▶ビット7＝エンド・オブDMA

$\overline{EOP}$＝“L”，$\overline{DACK}$＝“L”のとき$\overline{IOR}$，$\overline{IOW}$のどちらかが“L”の状態が最低100ns続いたときに，1にセットされます．$\overline{EOP}$は，最後のデータ・バイトがアウトプット・データ・レジスタに書き込まれたときに，DMACがアサートするので，SCSIバス上で，その最後のデータ・バイトが本当に転送されたかどうかは，$\overline{REQ}$と$\overline{ACK}$の信号をモニタして確かめなければなりません．

このビットは，モード・レジスタのDMAモード・ビットが0のときには常に0です．

▶ビット6＝DMAリクエスト

DRQ（ピン10）の状態をCPUがモニタするときに使用できます．DRQ信号は，$\overline{DACK}$＝“L”にするか，モード・レジスタのDMAモード・ビットを0にすることで，クリアされます．また，フェーズ・ミスマッチ・インタラプトが発生したときには，DRQはリセットされません．

▶ビット5＝パリティ・エラー

このビットは，モード・レジスタのイネーブル・パリティ・チェック・ビットが1にセットされ，データ・レシーブ中およびセレクション/リセレクション中にパリティ・エラーが発生したときに，1にセットされます．リセット・パリティ/インタラプト・レジスタを読み込むことによって，クリアされます．

▶ビット4＝インタラプト・リクエスト・アクティブ

インタラプトが発生していることを示すビットで，IRQ（ピン9）の状態をモニタします．リセット・パリティ/インタラプト・レジスタを読み込むことでクリアされます．

▶ビット3＝フェーズ・マッチ

SCSIバス上の$\overline{MSG}$，$\overline{C/D}$，$\overline{I/O}$の状態と，ターゲット・コマンド・レジスタの下位3ビットの状態が一致していると，1にセットされます．このビットは，常にアップデートされており，チップがイニシエータとして設定されているときに意味をもちます．

▶ビット2＝ビジィ・エラー

モード・レジスタのモニタ$\overline{BSY}$ビットが1にセット

されているとき，$\overline{BSY}$が“H”になってはいけないときに“H”になると，1にセットされます．このエラーが発生すると，すべてのSCSIバスへの出力ドライバがディセーブルされ，モード・レジスタのDMAモード・ビットがクリアされます．

● **スタートDMAセンド・レジスタ（W）アドレス＝5**

DMAセンド動作（DMA→SCSIバス）を開始します．イニシエータ，ターゲットの両方で使用されます．このレジスタにライト・アクセスする前に，モード・レジスタのDMAモード・ビットが1にセットされていなければなりません．なお，書き込む値は意味をもちません．

● **スタートDMAターゲット・レシーブ・レジスタ（W）アドレス＝6**

ターゲットが，DMAレシーブ動作（SCSIバス→DMA）を開始します．

このレジスタにライト・アクセスする前に，モード・レジスタのDMAモード・ビットおよびターゲット・モード・ビットの両方が，共に1にセットされていなければなりません．書き込む値は意味をもちません．

● **スタートDMAイニシエータ・レシーブ・レジスタ（W）アドレス＝7**

イニシエータが，DMAレシーブ動作（SCSIバス→DMA）を開始します．このレジスタにライト・アクセスする前に，モード・レジスタのDMAモード・ビット＝1，ターゲット・モード・ビット＝0に設定されていなければなりません．書き込む値は意味をもちません．

● **リセット・パリティ/インタラプト・レジスタ（R）アドレス＝7**

このレジスタにリード・アクセスすると，バス/ステータス・レジスタのパリティ・エラー，インタラプト・リクエスト・アクティブ，ビジィ・エラーの各ビットがクリアされます．読み込まれた値は意味をもちません．

§4-3

PC9801用SCSIアダプタの製作

里　和政

PC9801用SCSIホスト・アダプタ・ボードの回路図を図1に示します．パラレルI/O(8255Aなど)ボードを作る程度で製作できます．

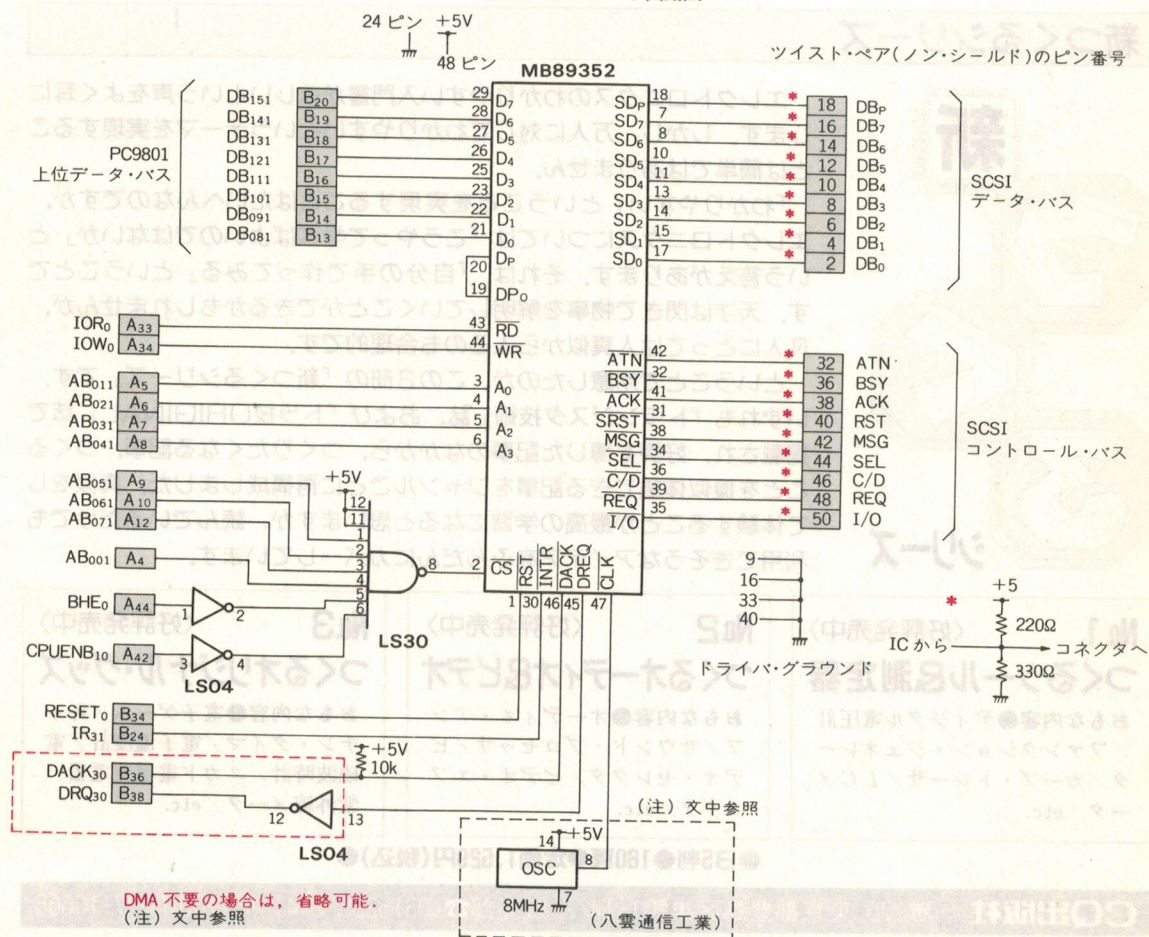
◆ ハードウェアの構成

コントローラのI/Oアドレスは，０Ｅ１Ｈ～０ＦＤＨまでを使用しました．PC9801では，これだけ広いI/O空間は少ないため，アドレスを変更する場合は，ほかのボードのI/Oと重ならないようにしてください．

また，DMAを使用する場合，PCの偶数のデータ・バス(DB_0～DB_7)に接続する必要があります．そのため，図1のコントローラのデータ・バスをPC9801の偶数のデータ・バスに接続します．このとき，I/Oアドレスは，０Ｅ０Ｈ～０ＦＥＨとなります．

コントローラからの割り込みは，8259A(マスタ)のINT_0(IR_{31})信号を使いました．ほかの割り込み信号

〈図1〉 SCSIボード回路図

DMA不要の場合は，省略可能．
(注) 文中参照

でも，とくに問題はないでしょうが，スレーブ側のときは，制御が複雑になります．

コントローラの入力クロックは，最大8MHzですから，PC9801から直接クロック入力する場合，CPUクロック($SCLK_1$)を8MHzにします．

48ピンのソケットが入手困難な場合，24ピンのソケットを2個並べることによって代用できます．

MB89352のドライバは，不平衡型(シングルエンド型)で終端抵抗が必要です．シンク電流は，48mA(0.5V DC)です．

SCSI用のコネクタは，50ピンのノン・シールド型のツイスト・ペア線を使用しました．

SCSIを使用したハード・ディスクの制御

開発用システムは，MS-DOSを使用し，デバイス・ドライバとして作成しました．MS-DOSのデバイス・ドライバには，標準ドライバと拡張ドライバがあります．

前者は，ＩＯ．ＳＹＳファイルに格納され標準装置(プリンタ，フロッピなど)がサポートされ，システムのローディングに自動的にロードされます．後者は，新しく追加したデバイスを使用したい場合に，容易にシステムに組み込むことができます(マウス，RAMディスク，拡張ハード・ディスクなど)．

◈ ハード・ディスクの仕様

今回使用したSEAGATE社の40Mハード・ディスクST251Nは，SCSIを標準インターフェースとしてもっています．表1にST251Nの仕様を示します．

SCSIでは，IDによって装置を識別するため基板にID設定用のディップ・スイッチがあり，標準はID＝0です．

コマンドは，グループ0，グループ1のCDB(コマンド・ディスクリプタ・ブロック)があります．図2にコマンドの基本形式を示します．

コマンドの内容はつぎのとおりです．

(1) グループ・コード：コマンドの記述形式を示す．ここでは，0または1．

(2) コマンド・コード：5ビットでコマンドを示す．

(3) LUN(論理ユニット番号)：ドライブのユニット番号．ここでは0．

(4) LBA(論理ブロック・アドレス)：MSB～LSBの21ビットでディスク上のブロック・アドレスを示す．グループ1では，32ビット．通常ディスクは，シリンダ，ヘッド，トラック，セクタによって区分けされているが，SCSI上ではすべて論理ブロックによって管理されているため，ブロックは連続している．アドレス0は，シリンダ，ヘッド，トラック，セクタのすべてが0である．

(5) 転送レングス：ブロックの転送数を示す．0の場合256ブロックの転送となる．グループ1では，最大65536ブロックの転送が可能となる．

(6) コントロール・バイト：リンク・ビットは，コマンドを連続して発行する場合“1”にする．フラグ・ビットは，リンク・コマンドが終了したときのステータス・メッセージのメッセージを示す．フラグ＝0のときＯＡＨ(LINKED-COMMAND COMP

〈表1〉[7] Seagate社ST251Nの仕様

項目	仕様
トラック数	3,272
シリンダ数	820
回転速度	3600±0.5%
ヘッド数	4
平均アクセス時間	40ms
ビット密度	14,902 BPI
記録方式	2.7 RLL
インターフェース	SCSI
容量フォーマット時	約40Mバイト
セクタ・サイズ	1024，512バイト

〈図2〉コマンド形式

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	コマンド・コード				
1	LUN			LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送レングス							
5	コントロール・バイト							

グループ・コマンド部は，0バイト目の7～5ビット

(a) 6バイト・コマンド

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	1	コマンド・コード				
1	LUN			—				REL
2	LBA (MSB)							
3	LBA							
4	LBA							
5	LBA (LSB)							
6	—							
7	転送レングス (MSB)							
8	転送レングス (LSB)							
9	コントロール・バイト							

(b) 10バイト・コマンド

バイト＼ビット	7	6	5	4	3	2	1	0
0	定義可		—				フラグ	リンク

(c) コントロール・バイト

LETE)，フラグ＝1のときOBH(LINKED-COMMAND COMPLETE With Flag)を送信する．

SASI仕様とはコマンドの異なる点があるため，注意が必要です．とくにPC9801で使用しているハード・ディスクの中には，インターフェース上ではSCSI仕様ですが，コマンド・レベルで異なる場合があります．

◈ MS-DOSデバイス・ドライバのプログラム

デバイス・ドライバを作成する場合，いくつかの規則があります．デバイス・ドライバの先頭0番地には，デバイス・ヘッダと呼ばれるヘッダがあり，ドライバのリンク・ポインタ，ストラテジ・エントリ・アドレス，割り込みエントリ・アドレス，およびデバイスの属性を記述します．

ストラテジ・エントリは，コマンド・ブロックのアドレスを受け取ります．コマンド・ブロックはコマンド・パケットとも呼ばれ，コマンド，パラメータの受け渡しに使用されます．

割り込みエントリは，コマンド・コードにしたがって，リード/ライトなどの処理を行います．

デバイスの属性は，ブロック，キャラクタなどを決定します．ハード・ディスクの場合は，ブロック型，NON FAT IDにします．

● BPB部

BPB(BIOSパラメータ・ブロック)は，ブロック・デバイスにおいてファイル構造を定義します．図3にBPBの構成を示します．

次にそのBPBの説明を示します．

(1) 1セクタのバイト数：論理1セクタのバイト数を示す．

DOSは，これを基準にしてセクタの管理を行います．

(2) 1ユニットあたりのセクタ数：DOSは，この値からユニットの大きさを決める．

CP/Mでは，データ・ブロックに相当します．1または複数のセクタから成り，2の累乗です．

大容量のディスクの場合この値が少ないと，ディスク全体をカバーできなくなります．

(3) 予備セクタ数：IPL，システムなどで使用するセクタ数を示す．不要な場合は0にします．

(4) FATの数：FAT(ファイル・アロケーション・テーブル)は，ユニットの使用状況を示すためのテーブルです．FATが破壊されるとファイルのアクセスが不可能となるため，信頼性を上げるために複数のFATを作成しておきます．通常は2個です．

(5) ルート・ディレクトリのエントリ数：この数は，ユニット・バイトを32(ディレクトリのバイト数)で割った値の倍数で決めます．

〈図3〉BIOSパラメータ・ブロックBPBの構成

フィールド	バイト数
1セクタのバイト数	2バイト
1ユニットあたりのセクタ数	1バイト
予備セクタ数	2バイト
FATの数	1バイト
ルート・ディレクトリ・エントリの数	2バイト
セクタの総数	2バイト
メディア・ディスクリプタ	1バイト
FATのセクタ数	2バイト

ここでいうセクタとは論理セクタを示す

(6) セクタの総数：全ディスクの論理セクタ数を示す．

(7) メディア・ディスクリプタ：メディアの交換を行えるデバイスの場合，交換したメディアのタイプを区別するためのもの．

ハード・ディスクの場合，メディアを変更することができないため，とくに指定する必要はありません．

(8) FATのセクタ数：FATのバイト数は，総セクタ数からユニットの総数を計算します．そのユニット数を1.5(ただしユニット数が4085以上であれば2とする)で割った値の2の累乗です．

◈ プログラミング

実際のプログラムにしたがって，コマンドの処理手順の説明をします．

● 初期化：デバイス・ハンドラをシステムに組み込んだのち，このエントリを実行します．SCSIコントローラMB89352の初期化を行います．

デバイスIDを“7”にし，アービトレーション・フェーズの許可をします．

次に，ディスクに対してテスト・ユニット・レディを出して，ディスクのレディ確認をおこないます．OKならユニット・カウントに1をセットします．

● データ・リード：指定されたセクタから指定バイト数のディスク・リードを行います．ディスク上の1セクタは，256バイトでフォーマットしているため，4セクタで論理1セクタとなります．リード中にディスク・エラーが発生したときは，リクエスト・ステータス・センス命令でエラー状態をリセットします．

● データ・ライト：指定されたセクタから指定バイト数のディスク・ライトを行います．その他は，リードと同様です．簡略化のためライト・ベリファイも同じにしています．

図4にディスク・コマンドを示します．

〈図4〉[6] ディスク・コマンド

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	フラグ	リンク

（1） テスト・ユニット・レディ命令

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	ステータス長							
5	0	0	0	0	0	0	フラグ	リンク

（3） リクエスト・センス命令

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0
1	0	0	0	LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送長							
5	0	0	0	0	0	0	フラグ	リンク

（5） リード命令

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	フラグ	リンク

（2） トラック0命令

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0
1	0	0	0	FMT DATA	CMP LST	リスト・フォーマット		
2	0	0	0	0	0	0	0	0
3	インターリーブ (MSB)							
4	インターリーブ (LSB)							
5	0	0	0	0	0	0	フラグ	リンク

（4） フォーマット命令

バイト＼ビット	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1	0
1	0	0	0	LBA (MSB)				
2	LBA							
3	LBA (LSB)							
4	転送長							
5	0	0	0	0	0	0	フラグ	リンク

（6） ライト命令

● **リクエスト・センス命令**

ステータス長：センスするステータスの長さを指定する
- 0：4バイト・ステータス型式
- 22：22バイト・ステータス型式 ｝使用しない
- 27：27バイト・ステータス型式 ｝使用しない

バイト＼ビット	7	6	5	4	3	2	1	0
0	VAL	エラー・クラス			エラー・コード			
1	LBA (MSB)							
2	LBA							
3	LBA (LSB)							

VAL：LBAが有効か

（a） 4バイト・ステータス型式

● **フォーマット命令**

FMT DATA
CMP LST
リスト・フォーマット ｝特定なトラックのフォーマットを行うときに使用する．

全トラック・フォーマットする場合は，すべて"0"．

インターリーブ：スキュと同様で，フォーマットするセクタ間隔を指定する．0でスキュー"1"．

各フェーズのコントロール手順

セレクト・フェーズは，ディスクIDをセットしてセレクト・コマンドを発行します．このとき自動的にアービトレーションが実行されます．

イン・アウト・フェーズは，転送フェーズにしたがってフェーズ・コマンドをコントローラにセットします．転送の方向は，フェーズによって実行します．図5のフローチャートを参照してください．

各コマンド実行後エラーが発生したときには，リクエスト・センス命令によってエラー状態を解除します（リスト1）．

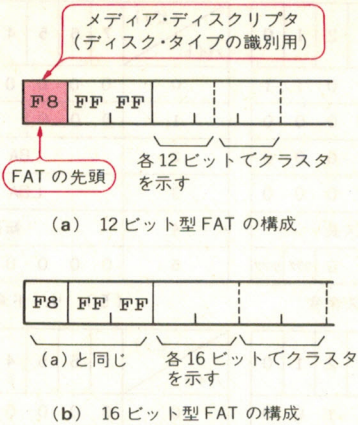
◆ ディスクのフォーマット

デバイス・ドライバがあっても，MS-DOS用にディスクがフォーマットされていないと実際に使用することはできません．物理的なフォーマットは，ディスクのフォーマット命令でおこないます．MS-DOSではFATとディレクトリ部のフォーマットだけで使用することができます．

図6にFATのフォーマットを示します．FATの先頭は，メディア・ディスクリプタがあり，これでディスクのタイプを識別します．ハード・ディスクでは，０Ｆ８Ｈとします．これが不正なIDであれば，

〈図6〉 FATの構成

(a) 12ビット型FATの構成

(b) 16ビット型FATの構成

メディア・ディスクリプタの種類

OF8H：ハード・ディスク
OF9H：640Kバイト・ディスク，1トラック9セクタ
OFBH：640Kバイト・ディスク，1トラック8セクタ
OFCH：160Kバイト・ディスク，1トラック9セクタ
OFDH：320Kバイト・ディスク，1トラック9セクタ
OFEH：256Kバイト・ディスク，1トラック26セクタ
160Kバイト・ディスク，1トラック8セクタ
1Mバイト・ディスク，1トラック8セクタ
OFFH：320Kバイト・ディスク，1トラック8セクタ

CHKDSKコマンドでエラーが発生します．

リスト2にフォーマットのプログラムを示します．このプログラムでのフォーマットは，フォーマット・コマンド標準の1セクタ256バイトにしています．

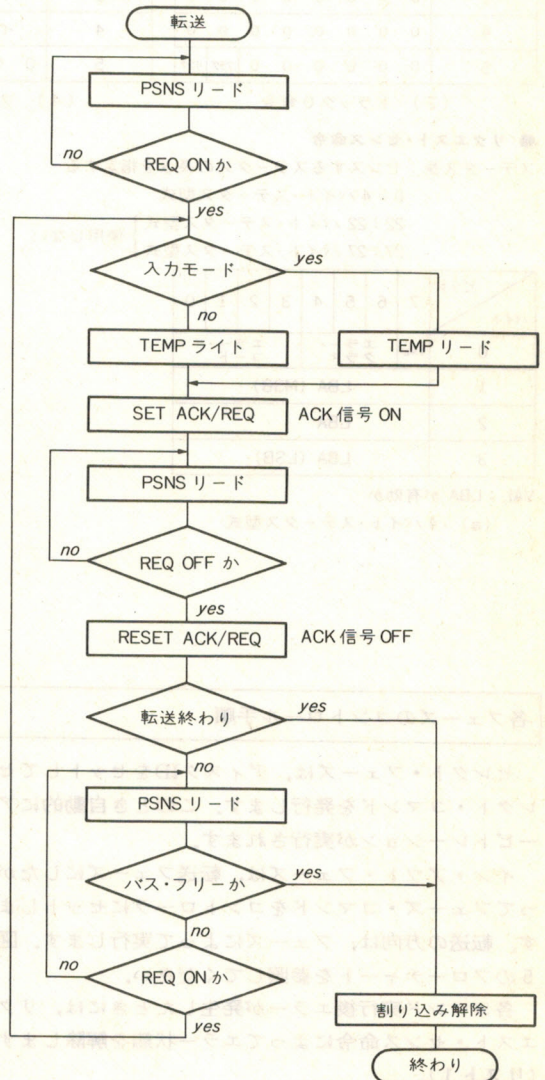
◈ リセレクトの手順について

SCSIは，前にも述べたように複数のデバイスを接続することができるため，1台のデバイスがバスを占有することは効率の低下のもとになります．そのために，デバイスがバスを使用する場合のみ占有することができます．

この方法は，デバイスによって異なる場合がありますが，SEAGATE社の仕様で説明します．

(1) セレクト時にATN信号をONにする．
(2) メッセージ・アウト・フェーズになり，リクエスト・メッセージとしてOCOHをターゲットに送信して，ディス・コネクトの要求を行う．
(3) コマンド・フェーズに移りターゲットにコマンドを送信する．
(4) コマンド受け付け後ターゲットは，ディス・コネクトを行うためメッセージ・イン・フェーズで04H(DISCONNECT)を送る．ただし，ステータス，メッセージ・イン・フェーズは存在しない．その後，バスは解放される．
(5) ターゲットがバスを使用する場合，ターゲットからアービトレーションを開始する．このときのイニシエータIDは，最初のセレクト時のイニシエータIDを使用する．
(6) ターゲットは，リセレクションを行い，リセレクション後メッセージ・フェーズで70Hを送る．
(7) 以後，通常処理と同じである．

〈図5〉 転送シーケンスのフローチャート

〈リスト1〉SCSIハード・ディスクの初期化, リード/ライト・プログラム

プログラムの使い方

(1) コントローラを初期化するために, SCSI_INITをコールする.
(2) データ・バッファ用のアドレスを, ESにセグメント, SIにオフセットを設定する.
(3) アクセスするセクタを, BX:CXに設定する.
(4) リードまたはライトするセクタ数をDLに設定する.
(5) ディスク・リードは, READを, ディスク・ライトは, WRITEをコールする
(6) リード, ライト時にエラーが発生した場合は, AXに0以外がセットされる.

```
;**************************************
;*
;*  scsi haed disk read/write program
;*
;*      ver 1.0
;*
;**************************************

data    segment

;*
;* scsi work area
;*
scsi_buff       equu    $
scsi_code       db      0       ; command code
scsi_lba0       db      0       ; block no.
scsi_lba1       db      0       ; block no.
scsi_lba2       db      0       ; block no.
scsi_len        db      0       ; block length
scsi_ctb        db      0       ; control btye
scsi_stat       db      0       ; status area
scsi_msg        db      0       ; message area
mode            db      0       ; i/o mode flag
req_stat        db      4 dup(?)

disk_buff       db      256*8 dup(?)

data    ends

CODE    SEGMENT byte

        ASSUME  cs:code,ds:data,es:data

;*************************
;
; D I S K   リード
;
;*************************
read:
        mov     ax,ds
        mov     es,ax
        mov     si,offset disk_buff     ; データ・バッファ
        mov     bx,0
        mov     cx,0                    ; セクタ  0  リード
        mov     dl,8                    ; 8セクタ
        call    disk_read
        ret

;*************************
;
; D I S K   ライト
;
;*************************
write:
        mov     ax,ds
        mov     es,ax
        mov     si,offset disk_buff     ; データ・バッファ
        mov     bx,2
        mov     cx,0                    ; セクタ  512  ライト
        mov     dl,8                    ; 1セクタ
        call    disk_write
        ret

;*************************************
;*
;* S C S I   D I S K   インターフェース
;*
;*************************************
;*
;* disk read
;*
;*  E S : S I   =   リード・バッファ・アドレス
;*  B X         =   リード・セクタ   (M S B)
;*  C X         =   リード・セクタ   (L S B)
;*  D L         =   リード・ブロック数
;*
disk_read:
        mov     scsi_code,08h   ; read command
        and     bl,01fh         ; lun mask
```

〈リスト1〉 SCSIハード・ディスクの初期化, リード/ライト・プログラム(つづき)

```
        mov     scsi_lba0,bl      ; i/o block no. set
        mov     scsi_lba1,ch
        mov     scsi_lba2,cl      ; lba lsb set
        mov     scsi_len,dl       ; read length (block count)
        mov     scsi_ctb,0        ; control byte
        call    scsi_select
        jnz     disk_error        ; select error
        call    cmd_send          ;
        jnz     disk_error        ;
        mov     bx,si             ; offset set
        mov     ch,scsi_len
        shl     ch,1              ; len * 512
        xor     cl,cl
        mov     al,1              ; data in phase
        call    scsi_io           ; read sector
        jnz     disk_error        ;
disk_status:
        mov     bx,cs
        mov     es,bx
        mov     bx,offset scsi_stat
        mov     cx,1
        mov     al,3
        call    scsi_io           ; status read phase
        jnz     disk_error        ;
        mov     bx,offset scsi_msg
        mov     cx,1
        mov     al,7
        call    scsi_io           ; message in
        jnz     disk_error        ; error
        cmp     scsi_stat,0       ; normal end ?
        jz      disk_end
        call    disk_req_sen      ; error request sense
        jnz     disk_error        ;
disk_end:
        xor     ax,ax
        ret
disk_error:
        mov     ax,-1
        ret
;*
;* disk write
;*
;* ES:SI    =  ライト・バッファ・アドレス
;* BX       =  ライト・セクタ  (MSB)
;* CX       =  ライト・セクタ  (LSB)
;* DL       =  ライト・ブロック数
;*
disk_write:
        mov     scsi_code,0ah     ; write command
        and     bl,01fh           ; lun mask
        mov     scsi_lba0,bl      ; i/o block no. set
        mov     scsi_lba1,ch
        mov     scsi_lba2,cl      ; lba lsb set
        mov     scsi_len,dl       ; write length (block count)
        mov     scsi_ctb,0        ; control byte
        call    scsi_select
        jnz     disk_error        ; select error
        call    cmd_send          ;
        jnz     disk_error        ;
        mov     bx,si             ; offset set
        mov     ch,scsi_len
        shl     ch,1              ; len * 512
        xor     cl,cl
        mov     al,0              ; data out phase
        call    scsi_io           ; write sector
        jmp     short disk_status
;*
;* disk sense
;*
disk_sense:
        mov     scsi_code,00h     ; sense command
        mov     scsi_lba0,0
        mov     scsi_lba1,0
        mov     scsi_lba2,0
        mov     scsi_len,0
        mov     scsi_ctb,0        ; control byte
        call    scsi_select
        jnz     disk_error        ; select error
        call    cmd_send          ; command execute
        jnz     disk_error        ;
        jmp     disk_status
;*
;* disk request sense
;*
disk_req_sen:
        mov     scsi_code,03h     ; sense command
        mov     scsi_lba0,0
        mov     scsi_lba1,0
        mov     scsi_lba2,0
        mov     scsi_len,0
        mov     scsi_ctb,0        ; control byte
        call    scsi_select
```

```
        jnz     req_sen_err         ; select error
        call    cmd_send            ; command execute
        jnz     req_sen_err
        mov     ax,cs
        mov     es,ax
        mov     bx,offset req_stat
        mov     cx,4
        mov     al,1                ; data in phase
        call    scsi_io             ; status read
        jnz     req_sen_err
        jmp     disk_status
req_sen_err:
        jmp     disk_error
;
; scsi command send phase
;
cmd_send:
        push    es
        push    bx
        push    cx
        mov     bx,cs
        mov     es,bx
        mov     bx,offset scsi_buff
        mov     cx,6                ; 6 byte send
        mov     al,2                ; command phase
        call    scsi_io             ; command send
        pop     cx
        pop     bx
        pop     es
        ret
;********************************
;*
;* scsi control
;*
;********************************
;
BDID    EQU     0E1H
SCTL    EQU     BDID+2
SCMD    EQU     BDID+4
INTS    EQU     BDID+8
PSNS    EQU     BDID+10
SDGC    EQU     PSNS
SSTS    EQU     BDID+12
SERR    EQU     BDID+14
PCTL    EQU     BDID+16
MBC     EQU     BDID+18
DREG    EQU     BDID+20
TEMP    EQU     BDID+22
TCH     EQU     BDID+24
TCM     EQU     BDID+26
TCL     EQU     BDID+28
;
; select
;
scsi_select:
        mov     al,0
        out     tch,al
        out     tcm,al              ; no wait
        mov     al,3
        out     tcl,al
        mov     AL,81H              ; DISK DEVICE ID
        out     TEMP,AL             ; BUS ID
        mov     al,0
        out     pctl,al             ; select
        mov     AL,00100100B        ; select command
        out     SCMD,AL
        xor     cx,cx
wait_select:
        in      AL,ints             ;
        test    al,10h              ; select ok ?
        loopz   wait_select         ;
        jz      scsi_sel_error
        out     ints,al
        xor     ax,ax
        ret
scsi_sel_error:
        mov     al,00000100b        ; bus free
        out     scmd,al
        mov     ax,-1
        or      ax,ax
        ret
;
; scsi phase in out
;
; AL     =  SCSI転送フェーズ
; ES:BX  =  I/O バッファ・アドレス
; CX     =  I/O レングス
;
scsi_io:
        push    bp
        mov     bp,sp
        sub     sp,2                ; work area get
```

〈リスト1〉SCSIハード・ディスクの初期化，リード/ライト・プログラム(つづき)

```
        mov     [bp],cx         ; save move length
        and     al,07h          ; mask
        out     pctl,al         ; transfer phase set
        mov     mode,al         ; transfer mode set i/o
        mov     cx,0            ; time out counter
phase_chk:
        in      al,psns         ; scsi status read
        test    al,80h          ; req ?
        loopz   phase_chk       ; no
        jz      scsi_io_err     ; time out
mode_chk:
        test    mode,1          ; input phase ?
        jnz     in_phase        ; yes
        mov     al,es:[bx]      ; data byte get
        inc     bx
        out     temp,al         ; send
        jmp     short set_ack
; input phase
in_phase:
        in      al,temp         ; data input
        mov     es:[bx],al
        inc     bx              ; data set
set_ack:
        mov     al,0e0h         ; set ack/req
        out     scmd,al
        mov     cx,0
out_wait2:
        in      al,psns
        test    al,80h          ; req ?
        loopnz  out_wait2       ;
        jnz     scsi_io_err     ; time out
        mov     al,0c0h         ;
        out     scmd,al         ; reset ack/req
        dec     word ptr [bp]   ; move counter down
        jz      phase_exit      ; end of data
        mov     cx,0
out_wait3:
        in      al,psns
        or      al,al           ; bus free ?
        jz      phase_exit      ; yes
        test    al,80h          ; req ?
        loopz   out_wait3       ; no
        jz      scsi_io_err     ;
        jmp     short mode_chk  ;
phase_exit:
        in      al,ints
        out     ints,al         ; intr reset
        xor     ax,ax
        jmp     short scsi_exit
scsi_io_err:
        mov     ax,-1
        or      ax,ax
scsi_exit:
        pop     cx              ; work area relese
        pop     bp
        ret
;********************************
; scsi controller init
;********************************
scsi_init:
        mov     al,7            ; BUS DEVICE ID
        out     BDID,al
        xor     al,al           ; RESET
        out     SCMD,al
        out     PCTL,al
        out     TCH,al
        out     TCM,al
        out     TCL,al
        out     TEMP,al
        mov     al,10000000B    ;
        out     SCTL,al         ; RESET
        mov     al,00010000B    ; ARBITRATION ENABLE
        out     SCTL,al
        ret
code    ends
        end
```

```
;***************************************
;*
;*  scsi head disk format program
;*
;*      ver 1.0
;*
;***************************************

CODE     SEGMENT byte

         ASSUME  CS:CODE,ds:code,es:code,ss:code
;*
;* scsi work area
;*
scsi_buff       equ     $
scsi_code       db      0       ; command code
scsi_lba0       db      0       ; block no.
scsi_lba1       db      0       ; block no.
scsi_lba2       db      0       ; block no.
scsi_len        db      0       ; block length
scsi_ctb        db      0       ; control btye

scsi_stat       db      0       ; status area
scsi_msg        db      0       ; message area
mode            db      0       ; i/o mode flag
sec_cnt         dw      0
;***************************************
;*
;* S C S I   D I S K  インターフェース
;*
;***************************************
;*
;* disk format
;*
start:
        mov     ax,cs
        mov     ds,ax
        mov     es,ax
        call    scsi_init
        call    disk_format
        mov     di,offset write_buff
        mov     cx,1024
        xor     al,al
        rep     stosb                   ; clear
        mov     bx,2                    ; next
        mov     cx,108                  ; fat & dir clear
null_write:
        push    cx
        push    bx
        mov     cl,2
        call    disk_write
        pop     bx
        pop     cx
        add     bx,2
        loop    null_write
        mov     di,offset write_buff
        mov     al,0feh
        stosb                           ; media id
        mov     ax,0ffffh
        stosw
        mov     cx,1024-3
        xor     al,al
        rep     stosb                   ; firast fat
        mov     bx,0
        mov     cl,2                    ; 2 block (1024 write)
        call    disk_write
        mov     bx,8
        mov     cl,2
        call    disk_write
        mov     ah,04ch
        int     21h

disk_format:
        push    cx
        call    scsi_select
        pop     cx
        jnz     disk_error              ; select error
        mov     scsi_code,04h           ; format command
        mov     scsi_lba0,0             ;
        mov     scsi_lba1,0
        mov     scsi_lba2,0             ; interleave msb
        mov     scsi_len,5              ;            lsb
        mov     scsi_ctb,0              ; control byte
        call    cmd_send                ;
disk_status:
        push    es
        push    bx
        mov     bx,cs
        mov     es,bx
        mov     bx,offset scsi_stat
```

```
        mov     cx,1
        mov     al,3
        call    scsi_io         ; status read phase
        mov     bx,offset scsi_msg
        mov     cx,1
        mov     al,7
        call    scsi_io         ; message in
        pop     bx
        pop     es
        xor     ax,ax
disk_error:
        ret
;*
;* disk write all e5h fill
;*
;* cx : write length (block count)
;* bx : start sector
;*
disk_write:
        push    cx
        call    scsi_select
        pop     cx
        jnz     disk_error      ; select error
        mov     scsi_code,0ah   ; write command
        mov     scsi_lba0,0
        mov     scsi_lba1,bh
        mov     scsi_lba2,bl    ; lba lsb set
        mov     scsi_len,cl     ; write length (block count)
        mov     scsi_ctb,0      ; control byte
        call    cmd_send        ;
        shl     cl,1
        mov     ch,cl
        xor     cl,cl
        mov     bx,offset write_buff
        mov     al,0            ; data out phase
        call    scsi_io         ; write sector
        jmp     short disk_status
;
; scsi command send phase
;
cmd_send:
        push    es
        push    bx
        push    cx
        mov     bx,cs
        mov     es,bx
```

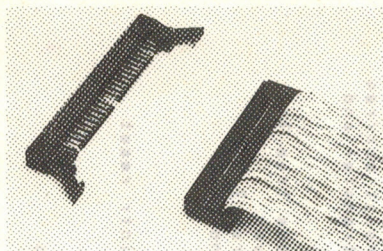
```
        mov     bx,offset scsi_buff
        mov     cx,6                    ; 6 byte send
        mov     al,2                    ; command phase
        call    scsi_io                 ; command send
        pop     cx
        pop     bx
        pop     es
        ret
atn_wait:
        in      al,psns
        and     al,07h
        cmp     al,6                    ; message out ?
        jne     atn_wait                ; no
        mov     al,40h
        out     scmd,al                 ; atn reset
        ret

;******************************
;*
;* scsi control
;*
;******************************
;
BDID    EQU     0E0H
SCTL    EQU     BDID+2
SCMD    EQU     BDID+4
INTS    EQU     BDID+8
PSNS    EQU     BDID+10
SDGC    EQU     PSNS
SSTS    EQU     BDID+12
SERR    EQU     BDID+14
PCTL    EQU     BDID+16
MBC     EQU     BDID+18
DREG    EQU     BDID+20
TEMP    EQU     BDID+22
TCH     EQU     BDID+24
TCM     EQU     BDID+26
TCL     EQU     BDID+28
;
; scsi controller init
;
scsi_init:
        MOV     AL,7                    ; BUS DEVICE ID
        OUT     BDID,AL
        XOR     AL,AL                   ; RESET
        OUT     SCMD,AL
```

〈リスト2〉SCSIハード・ディスクのフォーマット・プログラム(つづき)

```
        OUT     PCTL,AL
        OUT     TCH,AL
        OUT     TCM,AL
        OUT     TCL,AL
        OUT     TEMP,AL
        MOV     AL,10000000B        ;
        OUT     SCTL,AL             ; RESET
        MOV     AL,00010010B        ; ARBITRATION ENABLE
        OUT     SCTL,AL
        ret
;
; select
;
scsi_select:
        mov     al,0
        out     tch,al
        out     tcm,al              ; no wait
        mov     al,3
        out     tcl,al
        MOV     AL,81H              ; DISK DEVICE ID
        OUT     TEMP,AL             ; BUS ID
        mov     al,0
        out     pctl,al             ; select
        MOV     AL,00100100B        ; select command
        OUT     SCMD,AL
        xor     cx,cx
wait_select:
        IN      AL,ints             ;
        test    al,10h              ; select ok ?
        loopz   wait_select         ;
        jz      scsi_sel_error
        out     ints,al
        xor     ax,ax
        ret
scsi_sel_error:
        mov     al,00000100b        ; bus free
        out     scmd,al
        mov     ax,-1
        or      ax,ax
        ret
;
; scsi phase in out
;
; AL      =  SCSI転送フェーズ
; ES:BX   =  I/O バッファ アドレス
; CX      =  I/O レングス
;
scsi_io:
        and     al,07h              ; mask
        out     pctl,al             ; transfer phase set
        mov     mode,al             ; transfer mode set i/o
phase_chk:
        in      al,psns             ; scsi status read
        test    al,80h              ; req ?
        jz      phase_chk           ; no
mode_chk:
        test    mode,1              ; input phase ?
        jnz     in_phase            ; yes
        mov     al,es:[bx]          ; data byte get
        inc     bx
        out     temp,al             ; send
        jmp     short set_ack       ;
; input phase
in_phase:
        in      al,temp             ; data input
        mov     es:[bx],al
        inc     bx                  ; data set
set_ack:
        mov     al,0e0h             ; set ack/req
        out     scmd,al
out_wait2:
        in      al,psns
        test    al,80h              ; req ?
        jnz     out_wait2           ;
        mov     al,0c0h             ;
        out     scmd,al             ; reset ack/req
        jcxz    bus_free            ; transfer end
out_wait3:
        in      al,psns
        or      al,al               ; bus free ?
        jz      bus_free            ; yes
        test    al,80h              ; req ?
        jz      out_wait3           ; no
        jmp     short mode_chk      ;
bus_free:
        in      al,ints
        out     ints,al             ; intr reset
        ret
; buffer
;
write_buff      db      1024 dup(?)
                db      0
code    ends
        end     start
```

§4-4

NCR53C80評価用ボードの製作

清水哲夫

SCSIの動作および実際の転送速度などを調べるために，評価ボードを製作しました．主な仕様は以下の二つです．

(1) DMA転送可能なワンボード・マイコンとする
(2) SCSIポートには，市販されているSCSIコントロール用LSIを使用する

◈ CPUにはBASIC内蔵の8052を用いる

CPUには，インテル社の8052AH-BASICを用いました．このチップは，ワンチップ・マイコン8052の内蔵ROMに実数型BASICインタープリタをROMに焼き込んだものです．

今回のように，新しい周辺LSIの評価や，何かちょっとしたことをやりたいときにたいへん便利です．RS-232Cのダム・ターミナル(たれ流し端末)を接続するだけで，立派なBASICマイコンとなりますし，EP-ROM書き込みのソフトも内蔵しているので，書き込み電圧発生回路を付加すれば，作ったプログラムをその場でROMにできます．

また，パワーONで自動的にそのプログラムを起動させることもできるので，ターミナルを必要としない機器組み込み用のプロセッサとして随分と重宝しています．気になるのは，実行スピードと価格です．スピードに関しては，これはもうBASICということで，あきらめるしかなく，速度を要求する部分は，アセンブラのルーチンをBASICからCALLして何とかしのげます．

ただし，このオンチップBASICのもう一つの特徴に，BASICの機能の大部分(合計62個のサブルーチン)を，ユーザの作成するアセンブラ・ルーチンから，システム・コールという形で利用できるのです．この機能を利用することで，実数の四則演算はもとより，対数などの初等関数，ターミナルとのI/Oなど，全部をアセンブラで記述するには，かなりの仕事量となるものが，簡単にすませられます．使い方の例を図1に示します．

この機能をうまく使うと，BASICの部分は機械語サブルーチンへのCALL文一つで，残りはすべてそのアセンブラのルーチンですませるということも可能です．

価格は，秋葉原の店頭価格で約5000円と，Z80の約300円に比べれば20倍近い価格です．しかし，これもライブラリ・ソフト込みの値段と考えれば，それなりに納得がいくのではないかと思います．

● SCSIコントローラには53C80を使う

SCSIのコントロール用LSIには，NCR社の53C80を使いました．これは，SCSI用LSIとしては，おそらく世界で最初に量産されたと思われるNMOSタイプの5380のC-MOS版で，オープン・コレクタのドライバ/レシーバまで含んだ，ワンチップのコントローラです．

〈図1〉
BASIC内のライブラリの使用例

```
① MOV   A, #9AH  ;TOS=R2：R0
  CALL  30H      ;ライブラリ・コール
② MOV   A, #1FH  ;ルート演算  TOS=SQRT (TOS)
  CALL  30H
③ MOV   A, #1H   ;R3：R1=INT (TOS)
  CALL  30H
```

〔プログラムの説明〕
① R2：R0のレジスタ・ペアで示された16ビット整数を，実数に変換して，TOS（トップ・オブ・スタック）に入れる
② TOSの内容をルート演算してTOSに入れる（実数演算）
③ TOSの実数をポップして，16ビット整数に変換し，その値を，R3：R1のレジスタ・ペアに入れる

SCSI側のインターフェースとしての主な特徴は，

(1) バスの形態はシングル・エンド
(2) イニシエータ，ターゲットの両方をサポート
(3) データ転送速度は，最大1.5Mバイト/s
(4) アービトレーションをサポート
(5) パリティの生成およびチェックをサポート
(6) 同期転送はサポートしない

マイコン側のインターフェースとしての主な特徴は，

(1) プログラムI/O，DMAの両方共可
(2) マイコンへの割り込み可

ブロック図を図2に，全回路図を図3に示します．

回路の説明

● CPU

CPUは，最高スピードの12MHzで働かせます．これで80%のインストラクションは1μsのサイクル・タイムで動きます．

また，ここで用いたMAX232(マキシム社)は，TTLをRS-232Cレベルに変換するICで，DC-DCコンバータ内蔵のため，外付けコンデンサを四つつければ，5V単一電源で動作します．

アドレス・ライン(A_{15}〜A_0)と，データ・ライン(D_7〜D_0)および，WR，RD信号は，CPUのDMAACK信号により，フローティングにする必要があります．DMAのリクエストは，CPUのINT_0に接続します．というのも，このCPUは，DMA機能をハードウェアでサポートしているのではなく，ソフトウェアで処理しています．

すなわち，CPUはユーザからのDMAリクエスト信号を，割り込み入力のINT_0で受け付けると，その割り込み処理ルーチン内で，ポート1の第6ビット(DMAACK)を“L”にします．そして，自分自身は，INT_0信号が解除される(＝“H”)まで，ダミーのループをまわります．

ユーザ・デバイスは，DMAACK＝“L”によってフローティングになったアドレス，データ，R/W信号を，CPUに関係なく自由に使えるわけです．

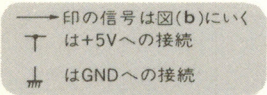
このDMAが通常のそれと違う点は，事実上バースト・モードしかできないので，DMAが何か仕事をしながら，同時にCPUのソフトが違う作業をすることができません．さらに，DMAアクノリッジのレイテンシ(リクエストからアクノリッジが返ってくるまでの時間)が割り込み入力を使うため，10μsのオーダになってしまうことです．

● メモリ

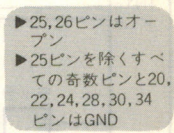
〈図2〉評価用ボードのブロック図

〈図3(a)〉 評価用ボードのCPU部の回路

→印の信号は図(b)にいく
┬ は+5Vへの接続
┴ はGNDへの接続

× 印は10kΩで+5Vへプルアップする

〈図3(b)〉 評価用ボードのメモリ,DMA,SCSI部の回路

▶25,26ピンはオープン
▶25ピンを除くすべての奇数ピンと20,22,24,28,30,34ピンはGND

×印は 220Ω 330Ω

メモリは，TC55257PL10が，BASICテキスト用のエリア(ＯＯＯＯH～７ＦＦＦH)で32Kバイト，M5M5165P10が，DMA転送用のバッファ(ＣＯＯＯH～ＤＦＦＦH)で，８Kバイト分用意してあります．

● DMAコントローラ

μPD8237AC5は，8085AファミリのDMAC(４チャネル)です．

Z80 DMAなどと違って，メモリ-メモリ間転送以外は，メモリまたはペリフェラル側が，データ・バスを直接駆動するタイプです．DMACは各々のリード/ライト信号(MEMR, MEMW, IOR, IOW)の制御に徹します．電源は＋５V ３A程度を用意します．

このワンボード・マイコンは，ターミナル(もちろんTERMモードにしたパソコンでもよい)を接続するだけで，BASICが動作します．ターミナルとの接続は，RxD, TxD, GNDの３本だけのRS-232Cですので，RS-232Cのほかのコントロール・ラインの制御を必要とするターミナルは注意が必要です．

また，XON/XOFFによるハンドシェイクも行えません．データ・ビット長は8，スタート，ストップ・ビットは共に1です．ボーレートは，オート・ボーレートかつプログラマブルですが，ハンドシェイクなしであることを考えると，2400bpsぐらいがちょうどよいかと思います．

動作の確認

製作が終了したら，いよいよ電源ONです．CPUは，RST信号がリリースされたあと，メモリがどれだけ実装されたかをチェックするルーチンを通ってから，ターミナルからのスペース・キー待ちとなります．

● ターミナルから動作をチェックする

ここでスペース・キーを１回たたくと，画面に次のようなメッセージが表示されます．

```
＊ＭＣＳ－５１(ｔｍ)ＢＡＳＩＣ　Ｖ１.１＊
ＲＥＡＤＹ
＞
```

最後の“＞”記号が，このBASICのプロンプトです．BASICは，最初のスペース・キーからボーレートを逆算して，ターミナルに合わせるわけです．

この後でボーレートを変更する必要が生じたときは，スペシャル・ファンクション・レジスタであるＲＣＡＰ２を，次の式に基づいて変更します．

$$\mathrm{RCAP2}=65536-\frac{\mathrm{XTAL}}{\mathrm{BAUD}\times 32}$$

ここで，XTALは使用した水晶振動子の周波数(単位はHz)で，このシステムでは12000000です．ＢＡＵＤは，変更したいボーレートです．したがって，ボーレートを9600に変えたいときは，

```
＞ＲＣＡＰ２＝６５４９７
```

とキー・インすれば，キャリッジ・リターンをたたいた瞬間に9600ボーに変わり，それからターミナルを9600に変えればよいのです．

● メモリのチェック

本システムでのメモリは，ＯＯＯＯH～７ＦＦＦHと，ＣＯＯＯH～ＤＦＦＦHにマッピングされます．BASIC上のシステム変数ＭＴＯＰは，０番地から連続して存在するメモリの最高アドレスを示します．

したがって，ＭＴＯＰをプリントすると，

```
＞ＰＲＩＮＴ　ＭＴＯＰ
　３２７６７
＞
```

となるはずです．

次に，ＣＯＯＯH～ＤＦＦＦHのメモリをチェックします．外部データ・メモリ空間にマッピングされたデバイスは，BASICのファンクションＸＢＹ(アドレス)で，リード/ライトできます．このデバイスは，CPUのWRとRD信号でストローブされた空間にあります．8052には，外部プログラム・メモリ空間というものもあり，こちらはCPUのPSEN信号によってストローブされるリード・オンリの空間のことです．

通常のBASICでは，ＰＥＥＫ，ＰＯＫＥに当たりますが，一つのオペレータで使えたほうが便利です．したがって，

```
＞ＰＲＩＮＴ　ＸＢＹ(ＯＣＯＯＯH)
　２５５
＞ＸＢＹ(ＯＣＯＯＯH)＝Ｏ
＞ＰＲＩＮＴ　ＸＢＹ(ＯＣＯＯＯH)
　Ｏ
＞
```

のような，ライト/ベリファイ作業を行うことで，メモリのチェックができます．これは何もメモリに限ったことでなく，外部データ・メモリ空間にマッピングされたデバイスは，すべてこの方法でリード/ライトできます．

これがことのほか便利で，筆者などは，とっかえひっかえ，マイコン周辺LSIと呼ばれるデバイスを接続し，この方法でチェックしています．まず，簡単なI/OプログラムをBASICで組んで基本動作を確認し，最後に，その部分を全部アセンブラにして高速化を図って，一件落着といったことをよくやっています．

なお，上の例で注意したいことは，16進数の書き表し方です．最後にHを付けるだけで16進数の意味になります．

(例)　１０H＝１６(10進)

ただし，最初のキャラクタが，A～Fで始まる16進数は，変数と区別するために，その上に０が必要です．

(例)　ＯＦ１H＝２４１(10進)

それで，C000Hを表すには，0C000Hと入力するわけです．

● **53C80のチェック**

イニシエータに設定して，SCSIバス上に$\overline{\text{RST}}$信号がでれば，OKということにします．

```
>XBY(0FE02H)=0
>XBY(0FE01H)=80H
```

最初の文で53C80をイニシエータに設定し，次の文でSCSIバス上の$\overline{\text{RST}}$を“L”にします．

ですから，ここで53C80の2番ピンが，“L”レベルになっていればOKです．

```
>XBY(0FE01H)=0
```

これで，$\overline{\text{RST}}$＝“H”にもどします．

ソフトウェア

ソフトウェアをリスト1に示します．プログラムの詳細は図4を参照してください．大部分は，SCSIコントローラである53C80のコントロールで占められています．53C80の詳しい使い方はLSIの説明をみていただくことにし，ここでは，DMAまわりについて説明します．

まずCPUの8052ですが，DMAを行うためには，次の二つのステートメントが必要です．

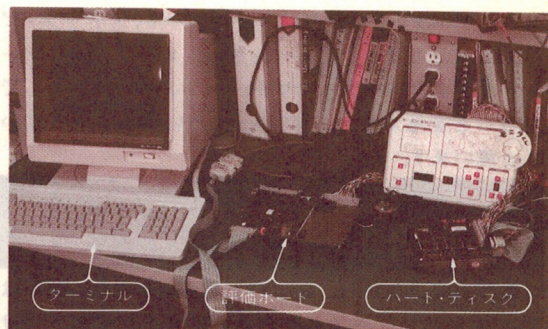
```
310 DBY(38)=DBY(38).OR.0
      2H
320 IE=IE.OR.81H
```

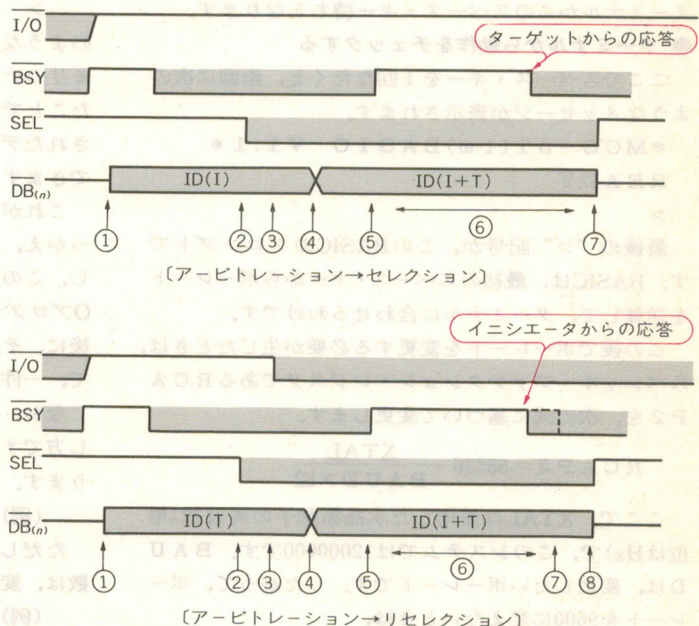
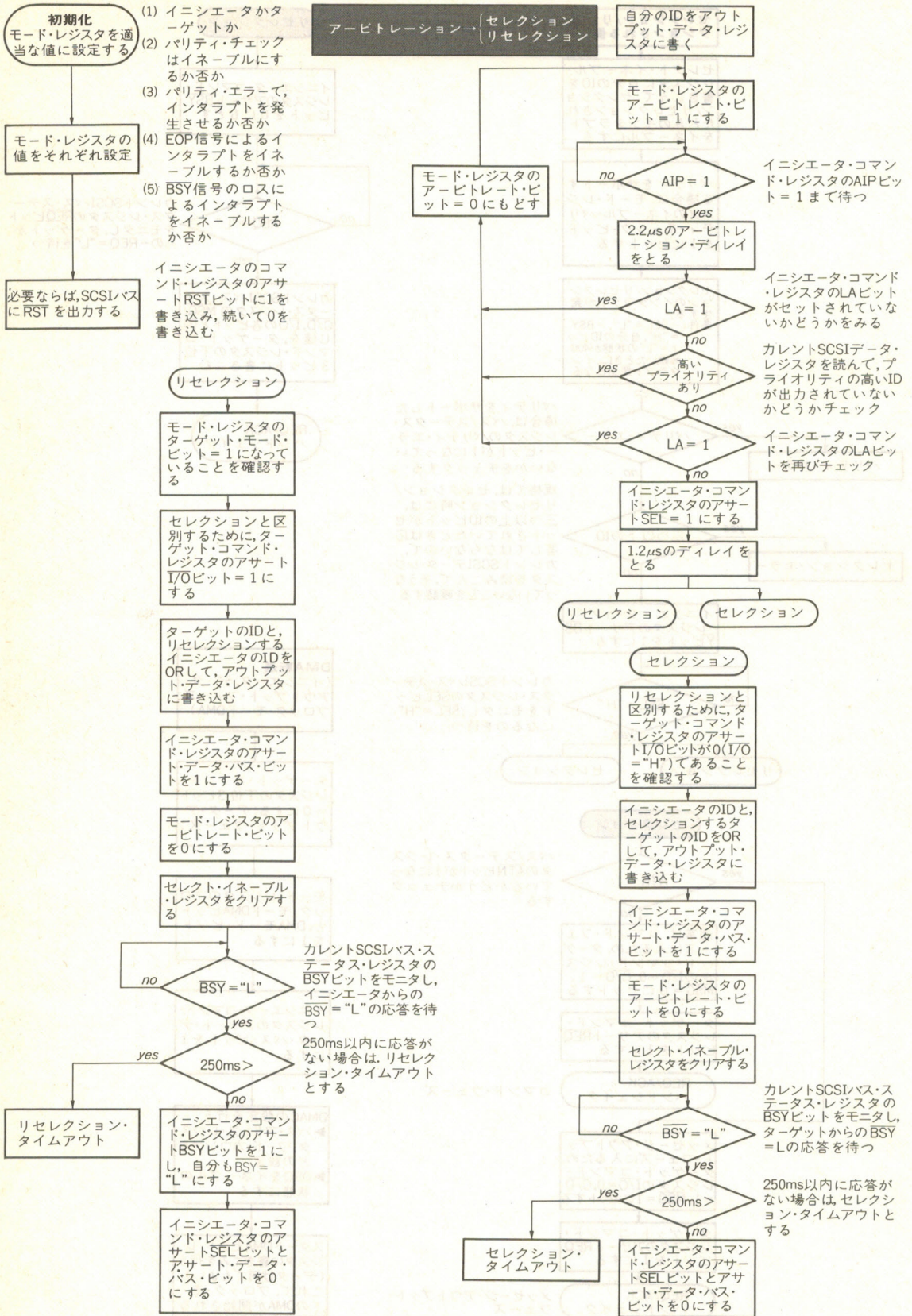
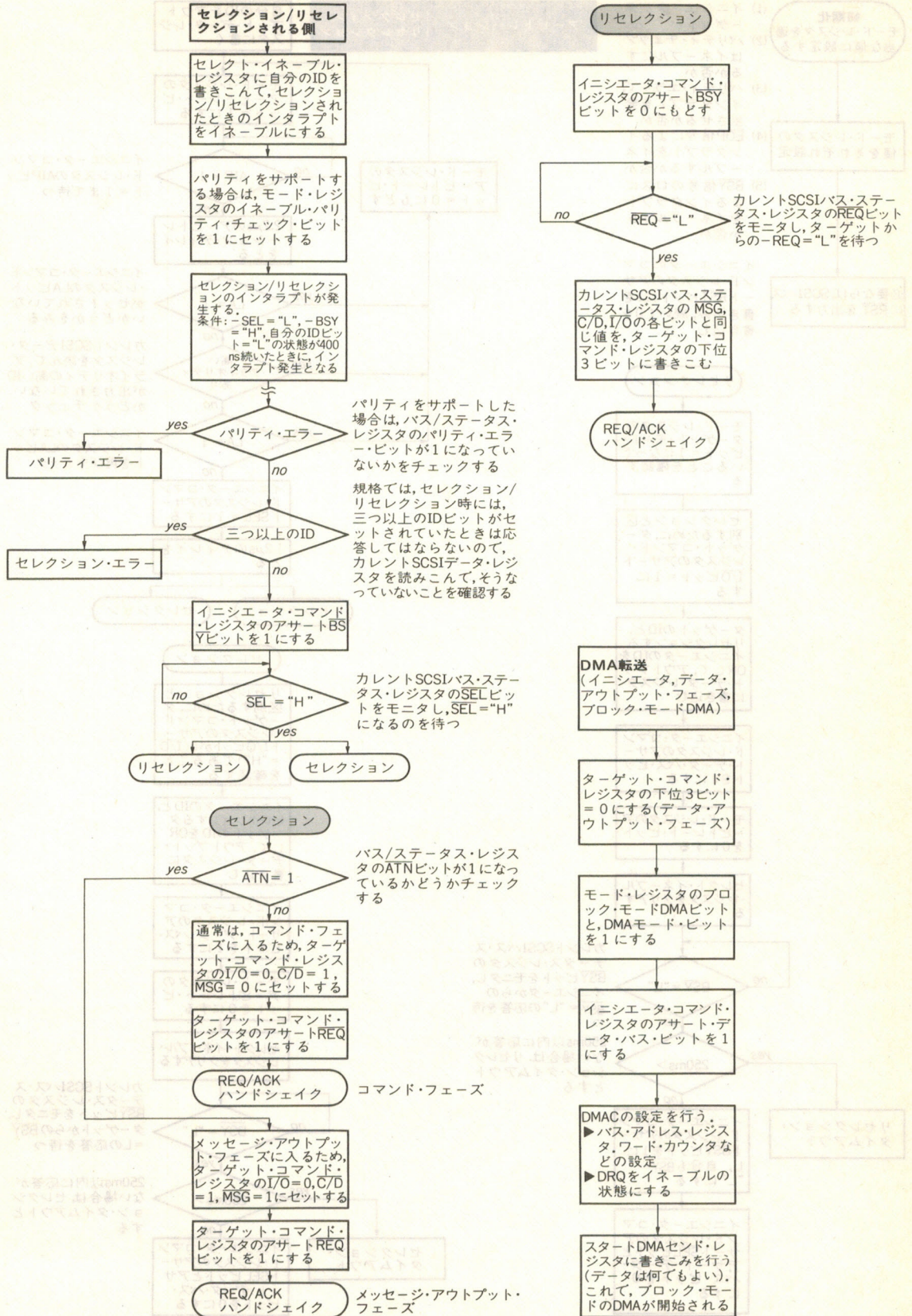
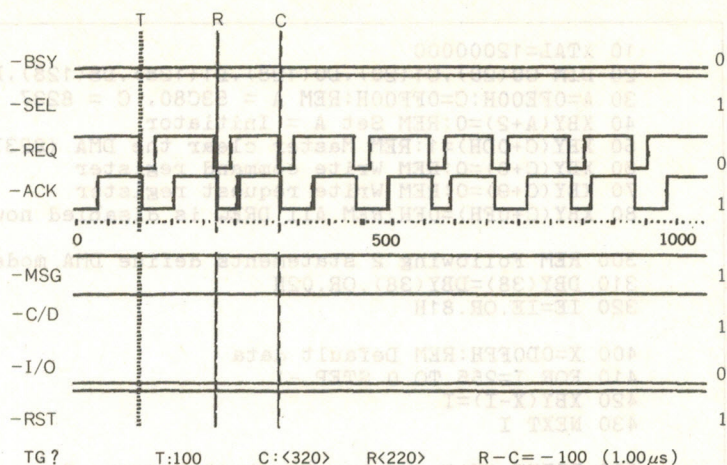
文番号310では，8052の内蔵RAMの38番地のビット1をセットします．これによって，INT_0からのインタラプトは，DMAREQの意味で解釈されて，7番ピンから外部のバッファなどを3ステートにする信号DMAACKが出力されます．

文番号320では，INT_0の割り込みをイネーブルにします．

接続実験

このワンボード・マイコンと，SCSIサポートの3.5インチ20Mバイト・ウインチェスタ・ディスク(ドライブはEPSON製，SCSIコントローラ・ボードはDTC製)を接続して，実際にSCSIバスの転送速度や，SCSIの使い勝手などを実験しました(**写真1**)．

本来ディスクに対してユーザが行おうとする動作は，リードとライトだけです．つまり，何番の論理ユニットの第何セクタから，何ブロック読み/書くという操作です．これ以外のことで必要なことといえば，ディスクのフォーマットができるということぐらいです．これらは，それぞれ，SCSIコマンド・グループ0の

〈写真1〉 接続実験のようす

〈図4(a)〉
リスト1の処理の詳細

〈図4(b)〉リスト1の処理の詳細

〈図4(b)〉 リスト1の処理の詳細(つづき)

〈図5〉
実際のタイミングを測定(データ・インプット・フェーズ，DMA転送)

READ(コード08H)，WRITE(0AH)，FORMAT UNIT(04H)で行うことができます．

実際，エラーが生じない限り，いったんフォーマットしてしまえば，二つのコマンドでディスクとやりとりできます．今までディスクの管理は，OSまかせにせざるを得なかったユーザも，OSには関係なく自分の管理下に置くことも容易に行えます．

また，転送速度もDMAのおかげで，1Mバイト/sでており(図5)，BASICで動いていることを忘れさせてくれるほどの快適さです．

◆参考・引用*文献◆

(1)*MB89352 技術資料 SPC-8704-01P，富士通．
(2)*MB89351 SCSIプロトコル・コントローラ 仕様書，富士通．
(3) PC-9801 ユーザーズ マニュアル，日本電気．
(4) PC-9801，MS-DOS3.1 ユーザーズ マニュアル，日本電気．
(5)*PC-9801，MS-DOS3.1 プログラマーズ リファレンス マニュアル，日本電気．
(6)*Seagate SCSI Interface Manual, November 20, 1986, 36021-001, Revision B.
(7)*ST251N/ST277N Product Manual, Rev A, Seagate，シーゲートジャパン．
(8) 清水哲夫；マイコン周辺LSI活用手引き書，第6回 SCSIの使い方〈1〉トランジスタ技術，1987年，5月号．
(9)*山本隆夫；SCSIコントローラLSIの動向，データム，1988年1月号，p.12，CQ出版社．
(10) ANSI X3.131-1986，SCSI規格書，日本規格協会．
(11)*5380-53C80 SCSI Interface Chip Design Manual, NCR (ジャパン マクニクス扱い)．
(12) John Katausky；MCS BASIC-52 USERS MANUAL, Intel.
(13) John Katausky；Built-in BASIC interpreter turns controller chip into versatile system core, Electronic Design, December 13, 1984, pp.175～182, Intel.
(14) 難波秀文；8052AH-BASICとワンボードマイコンの製作，プロセッサ，Nov 1985，pp.11～25，技術評論社．
(15) SCSIバスモニタ OZ101A 技術説明書，東陽テクニカ．
(16) DTC-310DB，OEMマニュアル，トーメンエレクトロニクス．
(17) マルチ・チップ，1986，pp.170～197，日本電気．

〈リスト1〉SCSIコントロール・プログラム

```
10 XTAL=12000000
20 DIM C0(20),C1(20),D0(128),D1(128),D8(128),D9(128)
30 A=0FE00H:C=0FF00H:REM A = 53C80, C = 8237
40 XBY(A+2)=0:REM Set A = Initiator
50 XBY(C+0DH)=1:REM Master clear the DMA (8237)
60 XBY(C+8)=0:REM Write command register
70 XBY(C+9)=0:REM Write request register
80 XBY(C+0FH)=0FH:REM All DREQ is disabled now

300 REM Following 2 statements define DMA mode
310 DBY(38)=DBY(38).OR.02H
320 IE=IE.OR.81H

400 X=0D0FFH:REM Default data
410 FOR I=255 TO 0 STEP -1
420 XBY(X-I)=I
430 NEXT I

500 PRINT "Hit any key to start program ",
510 X=GET:IF X=0 THEN GOTO 510
520 PRINT CR

600 REM**************************************************
610 REM      Reset
620 REM**************************************************
620 PRINT "Reset!"
630 XBY(A+1)=80H:XBY(A+1)=0:REM Assert RST*

1000 REM**************************************************
1010 REM      Arbitration phase
1020 REM**************************************************
1100 PRINT "Arbitration"
1110 INPUT "Enter initiator ID: ",X0:REM X0=2H
1120 XBY(A)=X0:REM Set ID bits in data out register
1130 XBY(A+2)=1:REM Start arbitration (BSY*=L, IDB(I))
1140 Z=XBY(A+1).AND.40H:IF Z=0 THEN GOTO 1140:REM Wait for AIP
1150 REM Here wait for 2.2 us and check if higher priority exists
1160 Z=XBY(A):IF Z<=X0 THEN 1170:REM Assume no higher priority exists
1170 XBY(A+1)=4:REM Now I'm winning, assert SEL*
1180 REM Here wait for 1.2 us

1400 REM**************************************************
1410 REM      Selection phase
1420 REM**************************************************
1500 PRINT "Selection"
1510 INPUT "Enter target ID: ",X1:REM X1=40H
1520 XBY(A+3)=0:REM I/O* = H for selection
1530 Y=X0.OR.X1:XBY(A)=Y:REM Prepare data bus --> IDB(I+T)
1540 XBY(A+1)=5:REM In order to keep SEL*=L and data bus
1550 XBY(A+2)=0:REM Now reset arbitration --> BSY*=H
1560 XBY(A+4)=0:REM Clear select enable register
1570 REM Now, wait for the X1 target assertion of BSY*
1580 REM And if no assertion within 250 ms, selection fails
1590 Z=XBY(A+4).AND.40H:IF Z=0 THEN GOTO 1590:REM Wait for BSY*
1600 XBY(A+1)=0:REM Deassert SEL* and DATA

2000 REM**********************************************
2010 REM      Now, check what phase the target requests
2020 REM**********************************************
2030 REM 1st, check for BF
2040 X=XBY(A+4).AND.42H:IF X=0 THEN GOTO 2100:REM BF detected
2050 REM Next, wait for REQ* comes
2060 X=XBY(A+4).AND.20H:IF X=0 THEN GOTO 2060
2070 GOTO 2200:REM Now, REQ* comes

2100 REM Here, detected BF
```

```
2110 PRINT "Bus Free":GOTO 500:REM Back to main portion

2200 X=XBY(A+4).AND.1CH:X=X/4
2210 ON X GOSUB 3000,4000,5000,6000,9000,9000,7000,8000
2220 GOTO 2000

3000 REM***********************************************************
3010 REM      Data Output phase
3020 REM***********************************************************
3100 PRINT "Data Output"
3110 INPUT "Pure data (1) or else (2): ",X
3120 IF X=1 GOTO 3300:REM Pure data
3130 INPUT "Enter # of bytes: ",Y
3140 FOR I=1 TO Y
3150 PRINT "Enter data byte: (",I,"):",
3160 INPUT " ",DO(I):NEXT I

3200 XBY(A+3)=0
3210 FOR I=1 TO Y
3220 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 3220:REM Wait for REQ*
3230 XBY(A)=DO(I):XBY(A+1)=1:REM Set data byte
3240 XBY(A+1)=11H:REM Assert ACK* and data bus
3250 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 3250:REM Wait for NOREQ
3260 XBY(A+1)=0:REM Deassert ACK* and data bus
3270 NEXT I
3280 RETURN
3300 XBY(A+3)=0
3310 XBY(C+0CH)=1:REM Clear 1st/last FF
3320 XBY(C+0AH)=0:REM Enable ch 0 DREQ
3330 XBY(C+0BH)=88H:REM Write mode register
3340 XBY(C)=0:XBY(C)=0D0H:REM Base address = 0D000H
3350 XBY(C+1)=0FFH:XBY(C+1)=0:REM Word count = 255
3360 XBY(A+2)=82H:REM Block mode DMA and DMA mode
3365 XBY(A+1)=XBY(A+1).OR.1
3370 XBY(A+5)=1:REM Now, start DMA Send
3380 REM Here, DMA should have been done
3390 PRINT "DMA finished":XBY(A+2)=0:XBY(A+1)=0:REM stop DMA
3400 RETURN

4000 REM***********************************************************
4010 REM      Data Input phase
4020 REM***********************************************************
4100 PRINT "Data Input"
4110 XBY(A+3)=1
4120 XBY(C+0CH)=1:REM Clear 1st/last FF
4130 XBY(C+0AH)=0:REM Enable ch 0 DREQ
4140 XBY(C+0BH)=84H:REM Write mode register
4150 XBY(C)=0:XBY(C)=0C0H:REM Base address = 0C000H
4160 XBY(C+1)=0FFH:XBY(C+1)=0:REM Word count = 255
4170 XBY(A+2)=82H:REM Block mode DMA and DMA mode
4180 XBY(A+7)=1:REM Now, start DMA receive
4190 REM Here, DMA should have been done
4200 PRINT "DMA finished":XBY(A+2)=0:REM stop DMA
4210 X=0C000H:FOR I=0 TO 255 STEP 16
4220 FOR J=0 TO 15
4230 PHO.XBY(X+I+J),
4240 NEXT J:PRINT CR
4250 NEXT I
4260 RETURN

5000 REM*******************************************************
5010 REM      Command phase
5020 REM*******************************************************
5100 PRINT "Command"
5110 INPUT "Enter # of bytes: ",Y
5120 FOR I=1 TO Y
```

〈リスト1〉SCSIコントロール・プログラム(つづき)

```
5130 PRINT "(",I,"):",
5140 INPUT " ",C0(I):NEXT I

5200 REM ----- Y = # of bytes, C0(I) = command data -----
5210 XBY(A+3)=2
5220 FOR I=1 TO Y
5230 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 5230:REM Wait for REQ*
5240 XBY(A)=C0(I):XBY(A+1)=1:REM Set data byte
5250 XBY(A+1)=11H:REM Assert ACK* and data bus
5260 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 5260:REM Wait for NOREQ
5270 XBY(A+1)=0:REM Deassert ACK* and data bus
5280 NEXT I
5290 RETURN

6000 REM*************************************************
6010 REM      Status phase
6020 REM*************************************************
6100 PRINT "Status"
6110 XBY(A+3)=3
6120 Z=XBY(A+4).AND.20H:IF Z=0 THEN GOTO 6120:REM Wait for REQ*
6130 S0=XBY(A):REM Read status byte
6140 XBY(A+1)=10H:REM Assert ACK*
6150 Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 6150:REM Wait for NOREQ
6160 XBY(A+1)=0:REM Deassert ACK*
6200 PH0. "Status byte = ",S0:RETURN

7000 REM ------------- Message Output -------------
7010 REM Message Output is not supported by DTC310DB

8000 REM*******************************************************
8010 REM      Message Input phase
8020 REM*******************************************************
8100 PRINT "Message Input"
8110 XBY(A+3)=7
8120 I=1:DO
8130  Z=XBY(A+4).AND.3CH:IF Z=3CH THEN GOTO 8300:REM MI and REQ*
8140  IF Z=1CH THEN GOTO 8130:REM Wait for REQ* in MI
8150  REM Here, MI phase is finished
8160  GOSUB 10000
8170  RETURN

8300  D1(I)=XBY(A):I=I+1:REM Read data
8310  XBY(A+1)=10H:REM Assert ACK*
8320  Z=XBY(A+4).AND.20H:IF Z=20H THEN GOTO 8320:REM Wait for NOREQ
8330  XBY(A+1)=0:REM Deassert ACK*
8340 WHILE I<=128
8350 FOR J=1 TO 128 STEP 16
8360 FOR K=1 TO 16
8370 IF J+K-1>128 THEN GOTO 8380 ELSE PH0. D1(J+K-1),
8380 NEXT K:PRINT CR
8390 NEXT J
8400 GOTO 8120

9000 REM ------------- IL -------------
9010 PRINT "Illegal phase!!!":STOP

10000 REM *** Print out other data bytes ***
10010 REM      Input = I, D1(1 - I)
10020 REM **********************************
10030 IF I=1 THEN RETURN:REM No other data available
10040 FOR J=1 TO I STEP 16
10050  FOR K=1 TO 16
10060   IF J+K-1>=I THEN GOTO 10090 ELSE PH0. D1(J+K-1),
10070  NEXT K:PRINT CR
10080 NEXT J:RETURN
10090 PRINT CR:RETURN
```

●本書掲載記事の利用についてのご注意 ── 本書掲載記事には著作権があり，また工業所有権が確立されている場合があります．したがって，個人で利用される場合以外は所有者の承諾が必要です．
　また，掲載された回路，技術，プログラムを利用して生じたトラブル等については，小社ならびに著作権者は責任を負いかねますのでご了承ください．

●ご質問はお手紙で ── 本書に関する技術的なご質問は，往復はがきか返信用封筒を同封した書簡で出版部あてにお寄せください．著者へ回送し，直接回答していただきます．質問の内容は当該記事を逸脱しない範囲で，できるだけ具体的に明記してください．また，電話やFAXによるご質問にはお応えできませんのであらかじめご了承ください．

トランジスタ技術 SPECIAL No.9

1988年5月1日 初版発行
1997年6月1日 第11版発行

発行人
編集人　蒲生良治

発行所　CQ出版株式会社　〒170 東京都豊島区巣鴨1-14-2
電　話　03-5395-2123(出版部)，03-5395-2141(販売部)
振　替　00100-7-10665

(定価は表四に表示してあります)

印刷・製本　三晃印刷株式会社

乱丁，落丁本はお取り替えします

Printed in Japan